VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra 456 – Katedra informatiky

Systém regulace teploty chladiče

System of Heatsink Temperature Control

2010 Tomáš Juřička

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tomáš Juřička**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Systém regulace teploty chladiče
System of Heatsink Temperature Control

Zásady pro vypracování:

Cílem práce je realizovat regulátor teploty chladiče elektronické součástky.

1. Navrhněte obvodové řešení systému regulace teploty chladiče.
2. Pro mikroprocesor vytvořte programové vybavení.
3. Systém zrealizujte ve formě funkčního prototypu.
4. Vytvořte úplnou technickou dokumentaci systému.

Seznam doporučené odborné literatury:

1. FUKÁTKO, J., FUKÁTKO, T., ŠINDELKA, J. *Teplo a chlazení v elektronice*. Praha : BEN, 1997. 30 s. ISBN 80-86056-24-4.
2. *PIC18F452* [online]. 2005 [cit. 2009-10-16]. Dostupný z WWW: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010296>.
3. LÁNÍČEK, Robert. *Elektronika*. Praha BEN 1998. 480 s.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radek Novák, Ph.D.**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA

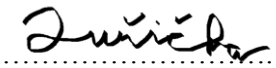
doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Čestné prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 7. 5. 2010



podpis studenta

Abstrakt

Cílem práce je realizovat systém regulace teploty chladiče elektronické součástky. Jednotlivé kroky jsou navrhnout obvodové řešení systému, vytvořit programové vybavení, realizovat systém ve formě funkčního prototypu a vytvořit technickou dokumentaci.

V práci je popsán postup při návrhu a následné realizaci regulátoru.

Klíčová slova

mikroprocesor, pic, dallas, teploměr, LCD, chladič, regulace, PID, ds18b20, pic16f877a, teplota

Abstract

The object of this task is implement a System of Heatsink Temperature Control. Propose a circuit solution system, implement software systém and construct prototype with technical documentation.

The paper describes the design and subsequent implementation of the controller.

Keywords

microcontroller, pic, Dallas, thermometer, LCD, radiator, control, PID, DS18B20, PIC16F877, temperature

Seznam použitých symbolů a zkratek

DPS – deska plošných spojů

PID – proporcionálně integrační derivační

PWM – pulzní šířková modulace

RB0 – port B, nultý bit

TMR0 – čítač/časovač 0

TMR1 – čítač/časovač 1

WDT – watch dog timer

Obsah

1. Úvod

Cílem práce je vytvořit funkční prototyp regulátoru teploty chladiče elektronické součástky. Jako mikroprocesor jsem zvolil pic16f877a, protože s ním mám již zkušenosti ze střední školy a následně i z vysoké školy.

Z počátku jsem práci programoval v assembleru, ale s přibývajícím kódem se program stával nepřehledným a nečitelným, proto jsem se rozhodl kód přepsat do jazyka C. Kompilátor jsem zvolil HI-TECH ANSI C Compiler ve verzi LITE. Je zdarma a pro mé účely naprosto dostačující. Základem regulátoru je již zmíněný mikroprocesor pic16f877a. Ke komunikaci s okolím jsem zvolil LCD display 2x16 znaků s vlastním řadičem. Teploměr jsem použil DALLAS ds18b20. Je digitální a komunikuje po jednom vodiči. Pro svou funkci potřebuje pouze napájení a jeden pull-up rezistor.

Chlazení/ohřívání zajišťuje peltierův článek, který je spínán přes výkonový unipolární tranzistor pomocí PWM signálu. Pro regulaci jsem zvolil PID algoritmus.

Dále jsou k dispozici 3 tlačítka. 1x reset a 2x tlačítko pro snižování a zvyšování požadované teploty.

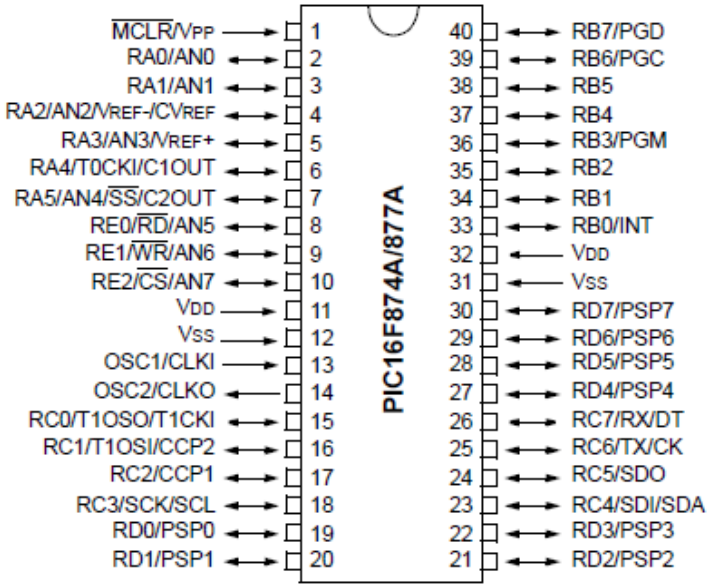
2. Hardware

2.1. Mikroprocesor pic16f877a

Tento mikroprocesor vyrábí firma MICROCHIP. Jelikož jeho instrukční sada má pouze 35 slov, je velice jednoduché naučit se programovat tento mikroprocesor. Může k němu být připojen až 20 MHz krystal → jeden instrukční cyklus trvá 200 ns. Parametry procesoru jsou uvedeny v příloze v tabulce č. 1. a 2.

Blokové schéma procesoru je v příloze na obrázku č. 1. Paměť pro data je rozdělena do čtyř bank. Banka se volí bity RP1:RP0. Mapa organizace paměti je zobrazena na obrázku č. 2. Speciální funkční registry jsou registry používané procesorem a přípojnými moduly pro nastavení požadovaného operačního módu. Tyto registry jsou implementovány jako statická RAM. Seznam speciálních funkčních registrů je v příloze v tabulce č. 3.

Pouzdro mikroprocesoru jsem zvolil 40-PIN PDIP. Rozložení pinů je na obrázku č. 1.



Obrázek 1: Pouzdro mikroprocesoru [1.]

2.1.1. Důležité funkční registry

Status Register:

STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

Tabulka 1: Status Register [1.]

IRP: Volba dvojstránky pro nepřímé adresování
1 = Bank 2, 3 (100h-1FFh)
0 = Bank 0, 1 (00h-FFh)

RP1:RP0: Volba stránky pro přímé adresování
11 = Bank 3 (180h-1FFh)
10 = Bank 2 (100h-17Fh)
01 = Bank 1 (80h-FFh)
00 = Bank 0 (00h-7Fh)

TO: Příznak přeplnění WDT
1 = Po připojení napájení, CLRWDT instrukci nebo SLEEP instrukci
0 = Došlo k přeplnění WDT

PD: Příznak napájení (instrukce „SLEEP“)
1 = Po připojení napájecího napětí nebo instrukcí CLRWDT
0 = Provedením instrukce SLEEP

Z: Příznak nulového výsledku operace
1 = Výsledek aritmetické nebo logické operace je „0“
0 = Výsledek aritmetické nebo logické operace není „0“

DC: Příznak přenosu z dolní do horní tetrády (ADDWF, ADDLW, SUBLW, SUBWF instrukce)
1 = Došlo k přenosu
0 = Nedošlo k přenosu

C: Příznak přenosu z byte (ADDWF, ADDLW, SUBLW, SUBWF instrukce)
1 = Došlo k přenosu
0 = Nedošlo k přenosu

OPTION_REG Register

OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

Tabulka 2: Option_REG Register [1.]

RBPU: Přiřazení vnitřních Pull-up rezistorů k PORTB
1 = Zakázány
0 = Povoleny

INTEDG: Výběr hrany pro přerušení na portu RB0
1 = Nástupná hrana
0 = Sestupná hrana

T0CS: Volba zdroje hodinového signálu pro TMR0
1 = RA4/T0CKI pin
0 = Vnitřní hodinový signál (CLKO)

T0SE: Výběr hrany pro inkrementaci TMR0
1 = Sestupná hrana na RA4/T0CKI pin
0 = Nástupná hrana na RA4/T0CKI pin

PSA: Přiřazení předděličky
1 = Přiřazena k WDT
0 = Přiřazena k TMR0

PS2:PS0: Volba předdělícího poměru

Hodnota bitů	Dělička k TMR0	Dělička k WDT
000	1:2	1:1
001	1:4	1:2
010	1.8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Tabulka 3: Volba předdělícího poměru [1.]

INTCON Register

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

Tabulka 4: INTCON Register [1.]

GIE: Globální povolení přerušení
1 = Povolí všechna nemaskovaná přerušení
0 = Zakáže všechna přerušení

PEIE: Přerušení od interních periferních obvodů
1 = Povolí všechna nemaskovaná periferní přerušení
0 = Zakáže všechna periferní přerušení

TMR0IE: Přerušení od TMR0 při přetečení
1 = Povoleno
0 = Zakázáno

INTE: Externí přerušení RB0/INT bit
1 = Povoleno
0 = Zakázáno

RBIE: Přerušení při změně na portu B
1 = Povoleno
0 = Zakázáno

TMR0IF: Příznak přetečení TMR0
1 = TMR0 registr přetekl, musí být nulováno softwarově
0 = TMR0 registr nepřetekl

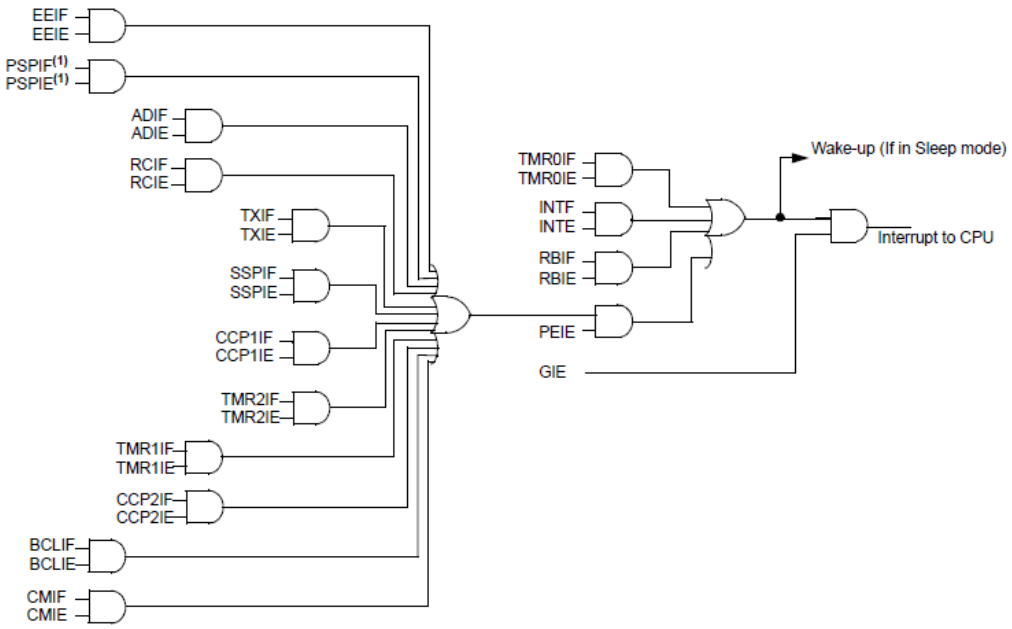
INTF: Příznak přerušení na RB0/INT
1 = Externí přerušení na RB0/INT obdrženo, musí být nulováno softwarově
0 = Nenastalo externí přerušení na RB0/INT

RBIF: Příznak přerušení na portu B
1 = Jeden z bitů RB7:RB4 se změnil, musí být nulováno softwarově
0 = Nezměnil se žádný z bitů RB7:RB4

Legenda:
R = Bit lze číst, W = Do bitu lze zapisovat U = Neimplementovaný bit, čtený jako ‘0’
- n = Hodnota po resetu ‘1’ = Bit je nastaven ‘0’ = Bit je vynulován x = Bit je neznámý
Ā – bit je aktivní v „0“

2.1.2. Přerušení

Mikropočítač má 15 zdrojů přerušení. Když dojde k vyvolání přerušení, tak dokončí rozdělanou instrukci a začne vykonávat program od adresy 0x04, dokud nenarazí na instrukci RETFIE. Potom se vrátí na místo, odkud skočil při vyvolání přerušení a pokračuje ve vykonávání kódu. Zdroj přerušení musí být zjištěn softwarově, protože všechny zdroje mají ten samý vektor přerušení. Příznak přerušení musí být nulován také softwarově. Logika přerušení je zobrazena na obrázku č. 2.



Obrázek 2: Logika zdrojů přerušení [1.]

2.1.3. Použité periférie

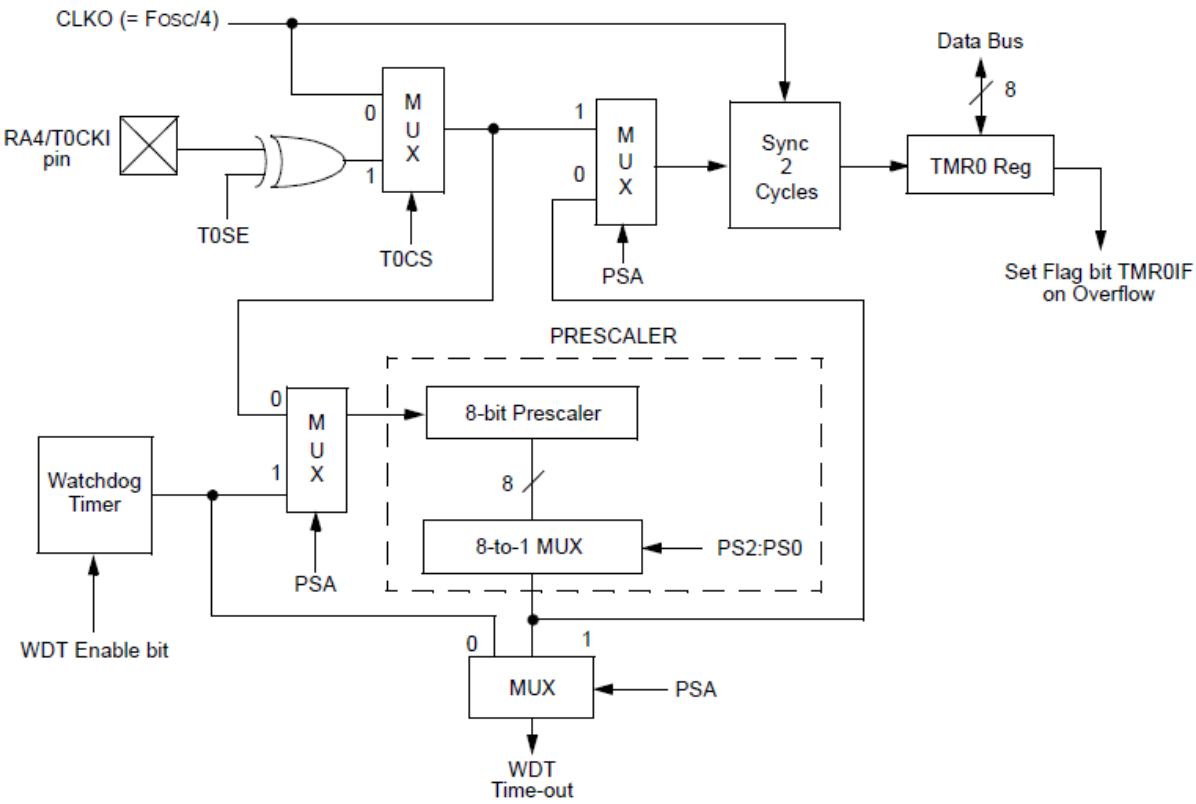
Porty:

Všechny porty se dají konfigurovat jako vstupní nebo výstupní. Některé piny mají alternativní funkci pro periferní obvody, jako například vstup pro A/D převodník, vstup hodinového signálu pro čítače apod.

Port B může mít přiřazeny interní pull-up rezistory, což usnadní připojení tlačítek. Já tuto funkci nevyužívám, protože při programování/krokování v aplikaci musí být tyto rezistory odpojeny. Navíc horní tetráda portu může sloužit jako zdroj přerušení, které je vyvoláno při změně hodnoty na portu.

Čítač/časovač 0:

Tento čítač je 8 bitový. Registr TMR0 může být inkrementován vnitřním hodinovým signálem ($f_{OSC}/4$), nebo signálem přivedeným na pin RA4/T0CKI. Přerušení je generováno, když hodnota registru TMR0 přeteče z hodnoty 0xFF na 0x00. Blokové schéma je zobrazeno na obrázku č. 3.



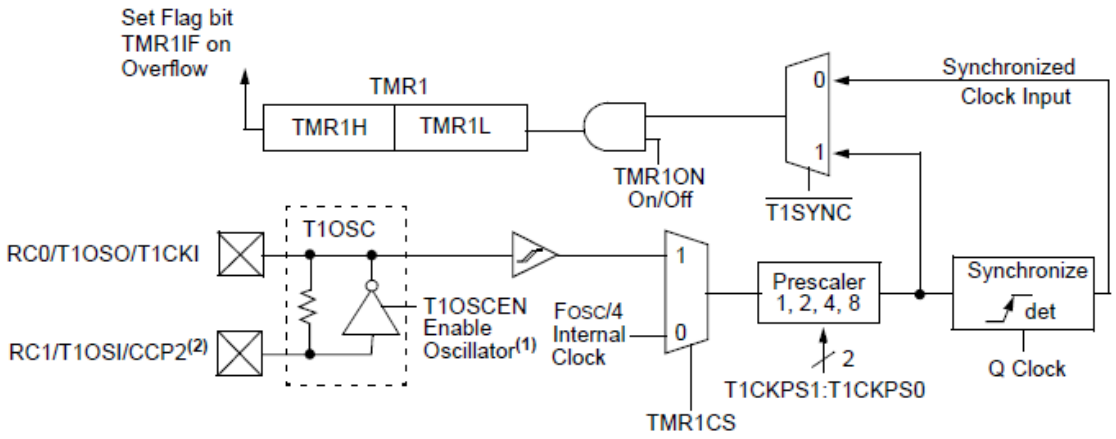
Note: T0CS, T0SE, PSA, PS2:PS0 are (OPTION_REG<5:0>).

Obrázek 3: Blokové schéma TMR0 [1.]

Registry INTCON a OPTION_REG jsou již popsány v kapitole **2.1.1 Důležité funkční registry.** V registru TMR0 je uložena aktuální načítaná hodnota.

Čítač/časovač 1:

Jedná se o 16 bitový čítač/časovač. Čítač může čítat, i když je procesor v režimu SLEEP a procesor následně „probudit“. TMR1 může být inkrementován vnitřním hodinovým signálem ($f_{OSC}/4$), nebo impulzy na pinu RC0. Přerušení je generováno při přetečení registru TMR1H:TMR1L z hodnoty 0xffff na 0x0000. Blokové schéma je zobrazeno na obrázku č. 4.



Obrázek 4: Blokové schéma TMR1 [1.]

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

Tabulka 5: TICON Register [1.]

— Neimplementováno, čteno jako ‘0’

T1CKPS1:T1CKPS0: Předdělička TMR1
11 = 1:8
10 = 1:4
01 = 1:2
00 = 1:1

T1OSCEN: Povolení oscilátoru TMR1
1 = Oscilátor povolen
0 = Oscilátor vypnut (invertor oscilátoru je vypnut pro snížení)

T1SYNC: Synchronizace vstupu externího hodinového signálu TMR1
Když TMR1CS = 1:
1 = Nesynchronizovat
0 = Synchronizovat
Když TMR1CS = 0:

TMR1CS: Zdroj hodinového signálu pro TMR1
1 = Externí zdroj na pinu RC0/T1OSO/T1CKI (nástupná hrana)
0 = Interní hodiny (FOSC/4)

TMR1ON: Zapnutí TMR1
1 = TMR1 zapnut
0 = TMR2 vypnut (nečítá impulzy)

V kapitole 2.1.bylo čerpáno z:[1.]

2.2.LCD display

Display jsem zvolil MC16021E8-SYL. Jedná se o LCD 2x16 znaků. Jedná se o display s řadičem HD44780. Parametry:

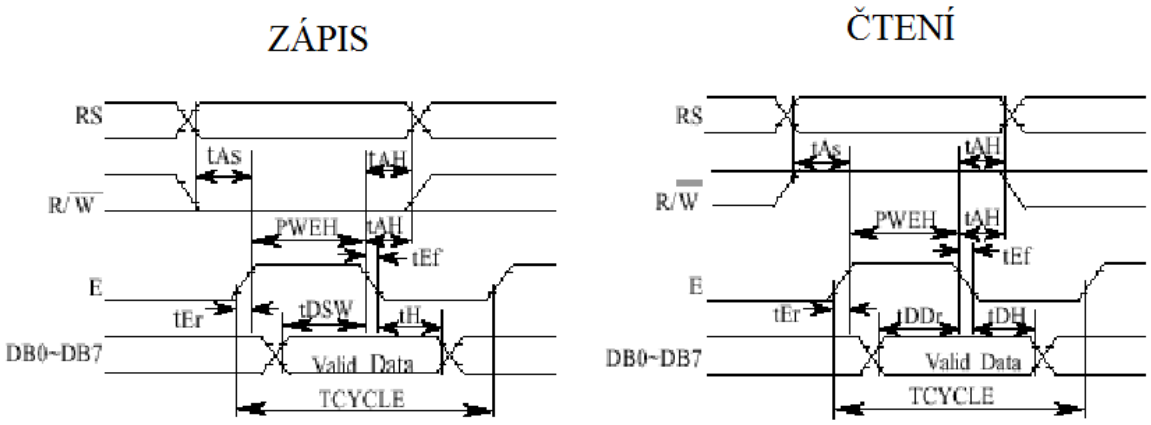
- Počet znaků: 32
- Velikost znaku: 4.86mm
- Napájecí napětí: 5V
- Display - šířka: 84mm
- Display - výška: 44mm
- Operační teplota: 0°C až +50°C
- Barva pozadí: Žlutá
- Technologie: STN
- Maximální napájecí napětí: 6V
- Rozlišeni znaku: 5 x 7 Teček + Kurzor

Display může komunikovat buď 4bitově + 3 řídící signály, nebo 8bitově + 3 řídící signály. Mnou zvolený procesor má dost vstupů a výstupů, proto nebyl důvod se zabývat 4bitovou komunikací a proto jsem zvolil 8bitovou.

PIN	Označení	Funkce
1	V_{ss}	0V (napájení)
2	V_{cc}	+5V (napájení)
3	V_0	Kontrast
4	RS	0 = vstup je instrukce 1 = vstup jsou data
5	R/W	0 = zápis dat do LCD 1 = čtení dat z LCD
6	E	Aktivace displeje
7	DB0	Data, bit 0
8	DB1	Data, bit 1
9	DB2	Data, bit 2
10	DB3	Data, bit 3
11	DB4	Data, bit 4
12	DB5	Data, bit 5
13	DB6	Data, bit 6
14	DB7	Data, bit 7
15		Nezapojen (podsvětlení)
16		Nezapojen (podsvětlení)

Tabulka 6: Piny LCD [4.]

2.2.1. Časové průběhy komunikace



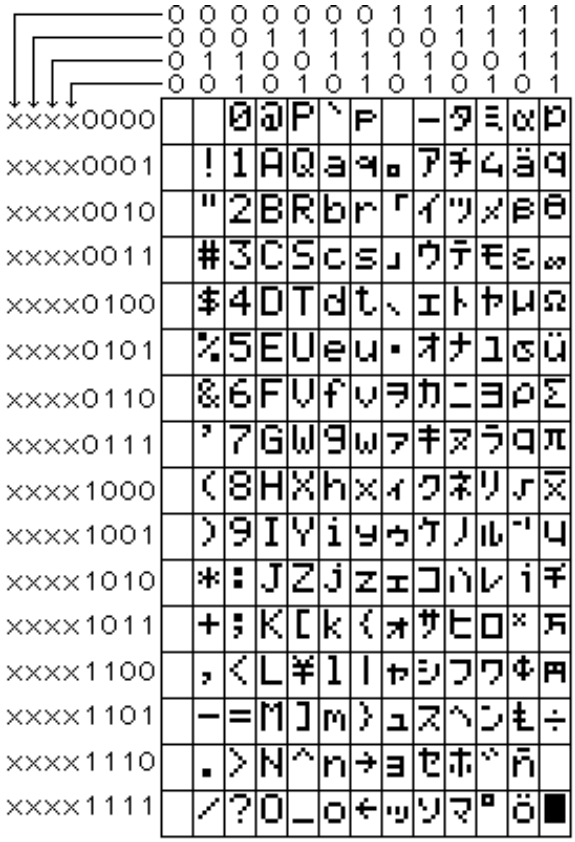
Obrázek 5: Časové průběhy [2.]

		STANDARD VALUE			UNIT
		MIN.	TYP.	MAX.	
Enable Cycle Time	t_{cycE}	1000	-	-	ns
Enable Pulse Width, Hight Level	PW_{EH}	450	-	-	ns
Enable Rise and Decay Time	t_{Er}, t_{Ef}	-	-	25	ns
Address Setup Time, RS, R/W-E	t_{AS}	140	-	-	ns
Data Delay Time	t_{DDR}	-	-	320	ns
Data Setup Time	t_{DSW}	195	-	-	ns
Data Hold Time (Write Operation)	t_{H}	10	-	-	ns
Data Hold Time (Read Operation)	t_{DHR}	20	-	-	ns
Address Hold Time	t_{AH}	10	-	-	

Tabulka 7: Vysvětlivky k obrázku č. 5. [2.]

2.2.2. Příkazy, znaky

U displeje 2x16 znaků mají jednotlivé pozice na displeji adresu v DDRAM 1. řádek 0x00h-0x0Fh, 2. řádek 0x40h – 0x4Fh. Znaková sada je zobrazena na obrázku č. 6. Adresa většiny znaků je shodná s ASCII tabulkou, ale například čísla jsou posunuta o hodnotu 30.



Obrázek 6: Znaková sada [4.]

Příkazy jsou shrnuty v příloze v tabulce č. 4. Postup inicializace displeje pro 8bitovou komunikaci:

- Počkat 15ms po přivedení napájecího napětí
- RS=0, R/W=0, DB=63
- Počkat minimálně 4,1 ms
- RS=0, R/W=0, DB=63
- Počkat minimálně 100 µs
- RS=0, R/W=0, DB=63

Po tomto příkazu se už může kontrolovat BF (Bussy flag – zda display ještě vykonává poslední příkaz), a nemusí se tedy měřit čas před dalším příkazem. Poté například:

- RS=0, R/W=0, DB=56 - Nastaví délku interface (8bit), počet řádků displeje (2) a znakový font (5x7 bodů).
- RS=0, R/W=0, DB=8 - Vypne displej, kurzor a jeho blikání.
- RS=0, R/W=0, DB=1 - Smaže displej a nastaví kurzor na pozici 0.
- RS=0, R/W=0, DB=6 - Určí směr pohybu kurzoru (Zvýšení pozice kurzoru) a posun displeje (display nepohybovat)
- RS=0, R/W=0, DB=12 - Zapne displej, kurzor a jeho blikání vypne.

V kapitole 2.2. bylo čerpáno z: [2.], [3.], [4.]

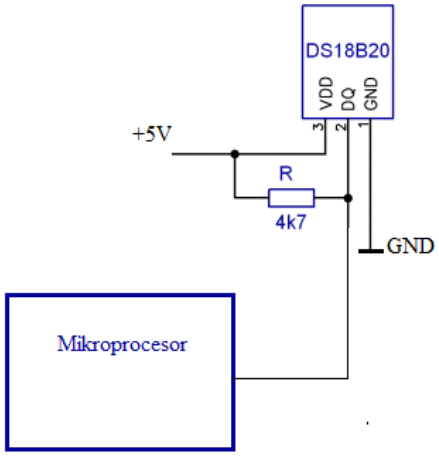
2.3.Teploměr dallas ds18b20

Jedná se o digitální teploměr s rozsahem -55°C až +125°C s přesností ±0.5°C mezi 10°C a +85°C. Rozlišení lze volit mezi 9 – 12 bity. Teplotu při 12bitové přesnosti změří za 750 ms. Napájecí napětí může být v rozmezí 3-5.5 V. Komunikuje prostřednictvím jediného vodiče pomocí 1-wire komunikačního protokolu. Připojit ho lze pouze 2 vodiči, a to GND a DATA input/output. Napájecí napětí může získávat z datového vodiče.

Na datový vodič musí být připojen pull-up odpor o hodnotě cca 5 kΩ. Na jednu sběrnici může být připojeno i více teploměrů nebo jiných zařízení komunikujících tímto protokolem. Identifikují se pomoci unikátní adresy, kterou má každý teploměr od výroby danou.

2.3.1. Komunikace

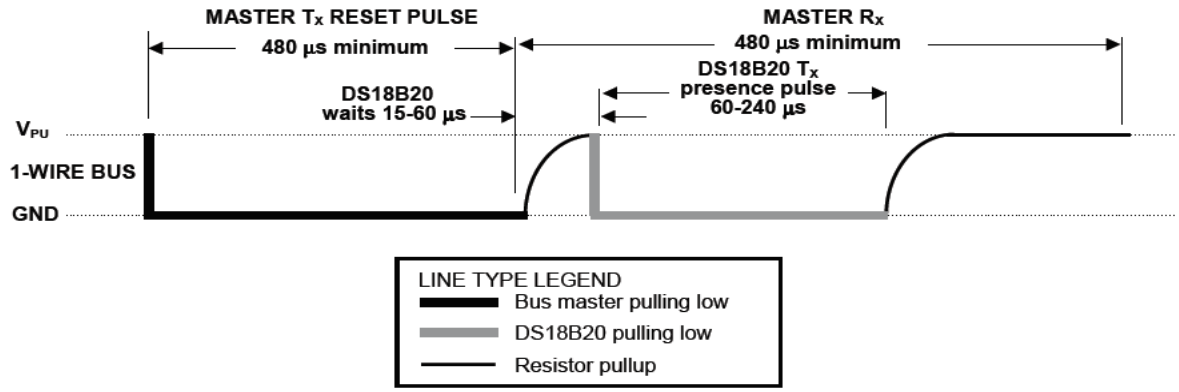
Schéma zapojení je zobrazeno na obrázku č. 7.



Obrázek 7: Schéma zapojení DS18B20 [6.]

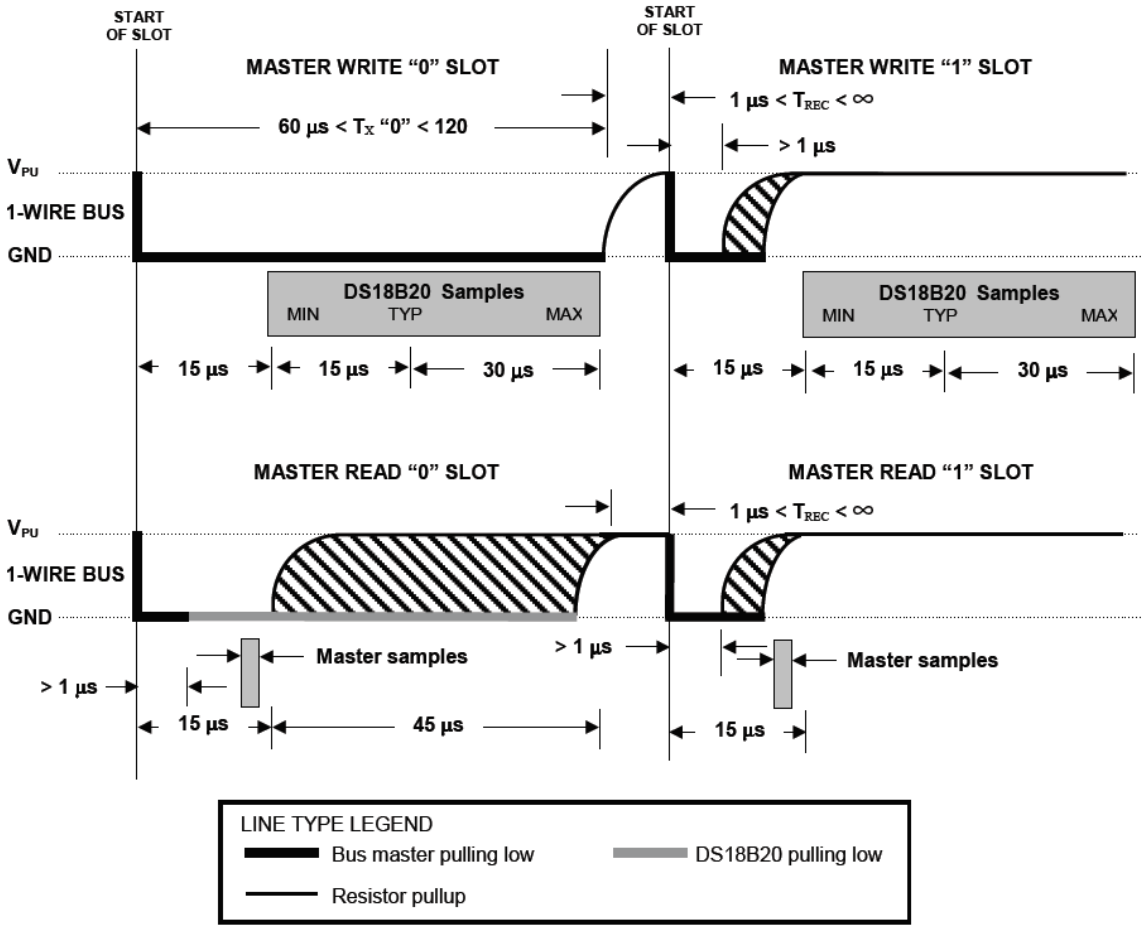
Mikroprocesor komunikuje s teploměrem prostřednictvím tzv. TIME SLOTŮ. Jeden slot trvá 60-120 µs a během něho je zapsán nebo přečten 1 bit. Komunikaci vždy zahajuje MASTER, v našem případě mikroprocesor, posláním reset pulzu. Poté, pokud je na sběrnici připojeno nějaké zařízení, odpoví mu stáhnutím sběrnice k logické „0“. Pokud se zařízení ohlásí, může master začít komunikovat.

Reset puls:



Obrázek 8: Resetovací pulz [6.]

Čtení/zápis:



Obrázek 9: Čtení/zápis bitů [6.]

Postup pro změření teploty:

- Vyslat reset puls
- Poslat 0xCCh – přeskočit identifikaci = pokud je na sběrnici pouze jedno zařízení, nemusí se sdělovat, s kým chceme komunikovat
- Poslat 0x044h – změř teplotu = odstartuje měření teploty
- Číst stav sběrnice – pokud teploměr ještě měří, tak stahuje sběrnici k „0“
- Vyslat reset puls
- Poslat 0xCCh – přeskočit identifikaci
- Poslat 0xBEh – chceme číst data – teploměr začne posílat data, která má uloženy v paměti, nejdřív pošle dolních 8 bitů, potom horních 8 bitů (bit s nejnižší váhou jako první), poté posílá ještě další informace, ale ty nás v této chvíli nezajímají.

Tento teploměr dokáže také hlídat rozmezí teplot. Můžeme mu nastavit horní a dolní mez, a pokud bude tato teplota překročena, tak po vykonání příkazu ALARM SEARCH (0xECh) teploměr stáhne sběrnici k „0“. Tahle funkce se hodí, pokud je na sběrnici více teploměrů, a my teplotu měříme a hlídáme na více místech.

V kapitole 2.3. bylo čerpáno z: [5.], [6.]

2.4.Peltierův článek

Peltierův článek funguje na základě Peltierova jevu, který objevil v roce 1834 Jean C. Peltier. Když prochází proud obvodem se dvěma rozdílnými vodiči zapojenými v sérii (většinou bismut a tellur), jedna z jejich styčných ploch se ochlazuje a druhá zahřívá.

Jejich hlavním nedostatkem je, že mají velkou spotřebu a nízkou účinnost. Nicméně mají také své výhody. Při přivedení napájecího napětí začne okamžitě jedna strana hřát a druhá chladit. Prostým otočením polarity lze prohodit chlazenou a ohřívanou stranu.

Při používání se teplá strana musí chladit, jinak hrozí, že dojde k přehřátí a pájka, kterou je peltier pospojován se rozteče.

Obrázek 10: Obrázek peltierova článku [7.]

Pro svou práci jsem zvolil peltierův článek M-TEC1-12705. Jeho parametry jsou (při teplotě okolí 27°C):

- I_{max} = 4.6 A
- ΔT_{max} = 68 °C
- U = 15.4 V
- Q_{max} = 33.4 W

V kapitole 2.4 bylo čerpáno z: [7.], [8.]

2.5.Ostatní součástky

2.5.1. Tranzistor IRFZ44N:

Jedná se o výkonový MOSFET. Zvolil jsem ho, protože ho lze spínat přímo mikropočítačem. Při 4,5 V je již dostatečně otevřen, aby sepnul Peltierův článek připojený na 12 V/4 A a nezahříval se.

2.5.2. Regulátor L7805CV:

Jedná se o regulátor v pouzdře TO-220. Jeho výstupem je konstantních 5 V/1 A, které požívám pro napájení mikropočítače, LCD, ventilátoru pro chlazení Peltierova článku a spínání relé.

2.5.3. Relé RELEH820F05C

Jedná se o dvojitý přepínač (AC 250V 5A, DC 30V 5A) ovládaný 5V/140 mA. Protože mikropočítač nezvládne na žádném portu 140 mA, spínám cívku relé pomoci MOSFET tranzistoru IRF630. Relé slouží k přepínání mezi módem topení a chlazení Peltierova článku.

3. Software

Program jsem tvořil ve vývojovém prostředí MPLAB IDE verze 8.36.00.00 doplněné o kompilátor HI-TECH ANSI C Compiler. Protože program je poněkud rozsáhlejší, rozhodl jsem se kompletní okomentovaný program umístit do příloh, a zde popsat detailněji jeho hlavní části, viz příloha zdrojový kód. Zdrojový kód bude také přiložen na CD, které je součástí této práce.

3.1.void interrupt preruseni(void)

Tato funkce je volána, dojde-li k vyvolání přerušení. Jako první se kontroluje, zda zdrojem přerušení není port B, pokud jím bylo, zjistí se, zda bylo zmáčknuto tlačítko nahoru, nebo dolů. U každého tlačítka se provede odpovídající akce. Volená teplota je shora omezena na 120 °C a zdola na -50 °C. Poté se spustí TMR1, který zajistí prodlevu cca 210 ms, aby se eliminovaly zákmity tlačítka, a tak nedošlo k několika změnám při jednom zmáčknutí.

Poté se kontroluje přerušení od TMR1. Dojde-li dvakrát k přerušení vyvolaném TMR1, povolí se opět přerušení od portu B, na kterém jsou připojena tlačítka.

Jako poslední se kontroluje přerušení od TMR0. TMR0 využívám jako časovač pro generátor PWM signálu. Zkoušel jsem použít PWM generátor integrovaný v mikroprocesoru, ale narazil jsem tam na problém se zdrojem.

K napájení výrobku používám spínaný počítačový ATX zdroj, a ten i při frekvenci 1.22 kHz vydával nepříjemné zvuky. Pomoci TMR0 generuji PWM signál o frekvenci cca 77 Hz, který je dostačující a zdroj s ním nemá problémy.

3.2.void pid( )

Tato funkce se stará o celou regulaci teploty. Ze zdroje [9.] jsem pochopil, co je principem PID regulace a co jednotlivé složky představují. Ze zdrojů [10.] a [11.] jsem se nechal inspirovat kódem pro regulaci, částečně jej upravil pro mé potřeby a použil jej.

- Err -chyba
- Tpoz- požadovaná teplota

- Tsku- skutečná teplota
- Kp-konstanta proporcionální složky
- Ilast-minulá integrační konstanta
- Ti – konstanta integrační složky
- Td – konstanta derivační složky
- T – vzorkovací čas
- ErrLast – předchozí chyba

$$Err = (Tpoz - Tsku)$$

3.2.1. P

Tato složka představuje prostý zesilovač, vezme rozdíl požadované teploty a skutečné teploty a tuto hodnotu zesílí.

$$P = Err * Kp$$

3.2.2. I

Integrační složka má za úkol předpovídat, jaká bude odchylka. Tato složka dokáže úplně odstranit regulační odchylku. Pokud je akční veličina delší dobu v saturaci tak se stává, že integrační složka neúměrně roste. Proto se při saturaci musí tato složka omezit. Tento jev se nazývá WIND-UP.

$$I=Ilast+(Kp*T/Ti)*Err$$

3.2.3. D

Derivační složka zrychluje regulační děj.

$$D= (Kp*Td/T)*(Err-ErrLast);$$

Výsledek získáme prostým sečtením všech tří složek. Konstanty PID složek jsem volil metodou Pokus-omyl popsanou ve zdroji [9.].

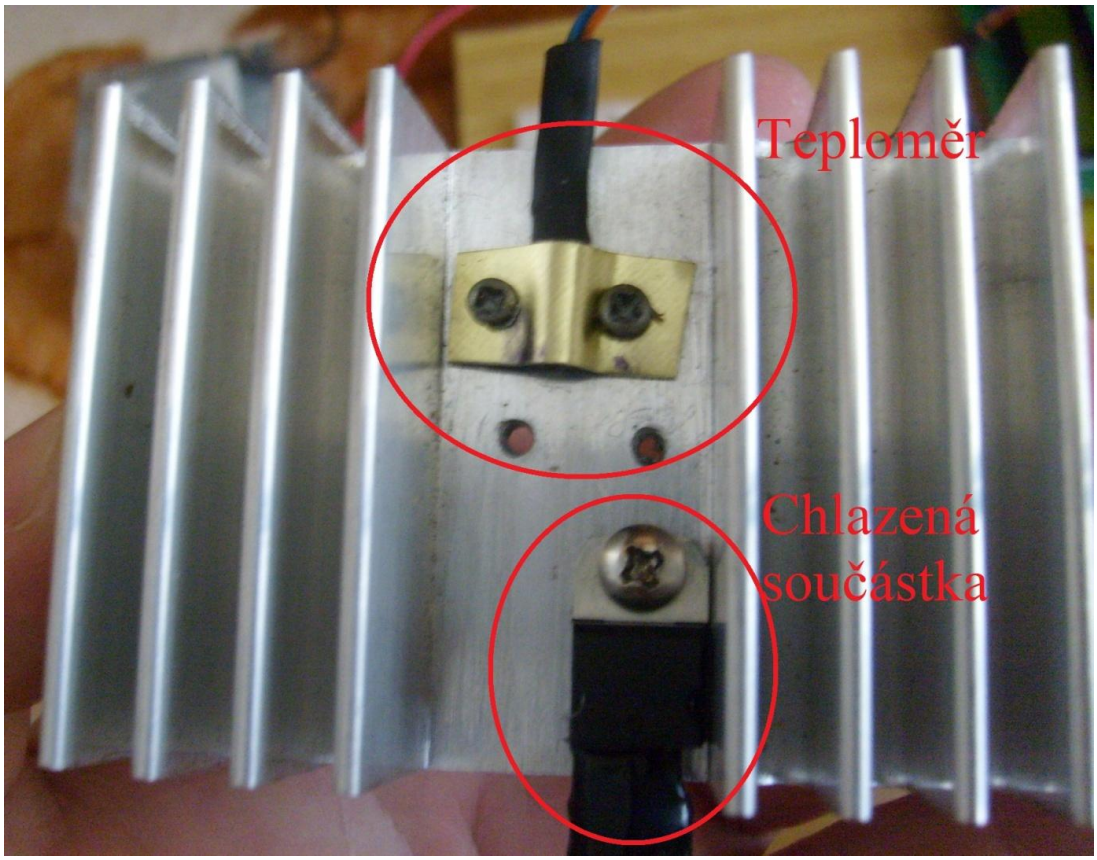
4. Prototyp

Schéma zapojení a DPS jsem navrhoval v programu Eagle 5.7.0 Light. Schéma zapojení je v příloze na obrázku č. 3., návrh desky z pohledu spojů v příloze na obrázku č. 6. a návrh desky z pohledu součástek v příloze na obrázku č. 5. Návrh DPS jsem přenesl na cuprextit a vyleptal v roztoku chloridu železitého.

Výsledný prototyp je zobrazen v příloze na obrázku č. 7. a 8.

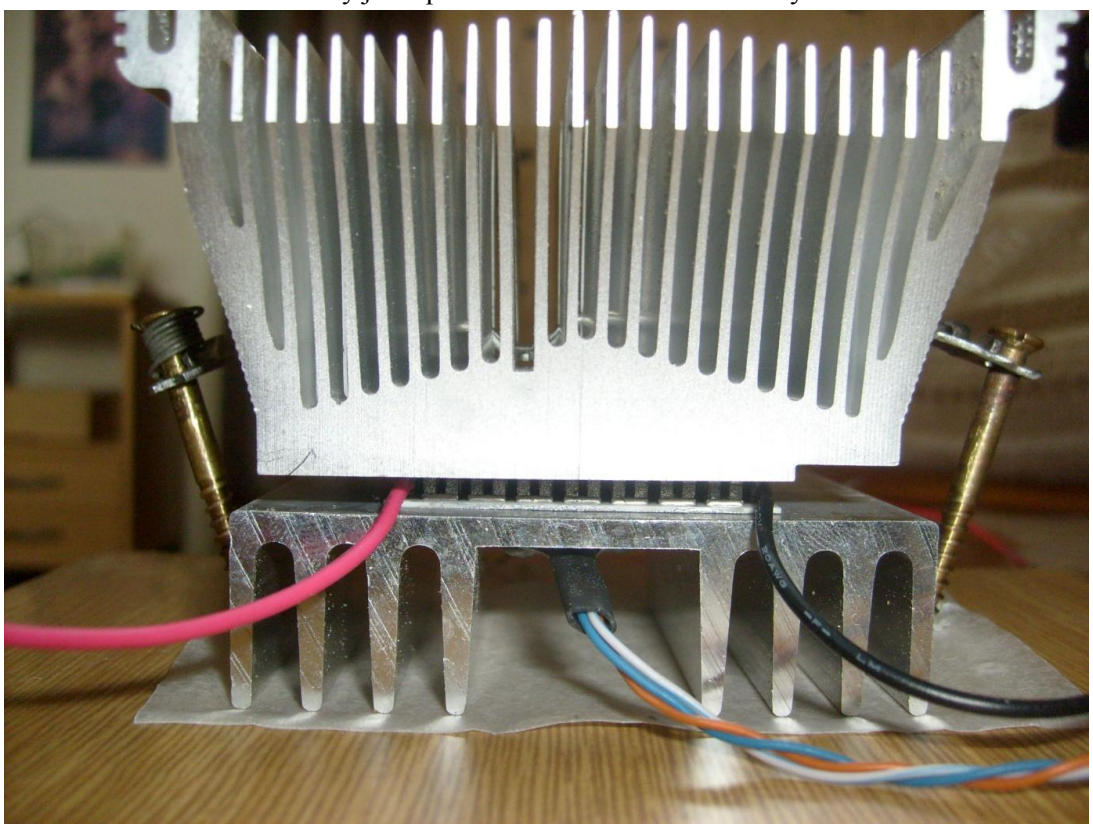
4.1.Konstrukce

Teploměr a chlazenou součástku (zdroj poruchy PID regulace) jsem přišrouboval k chladiči.



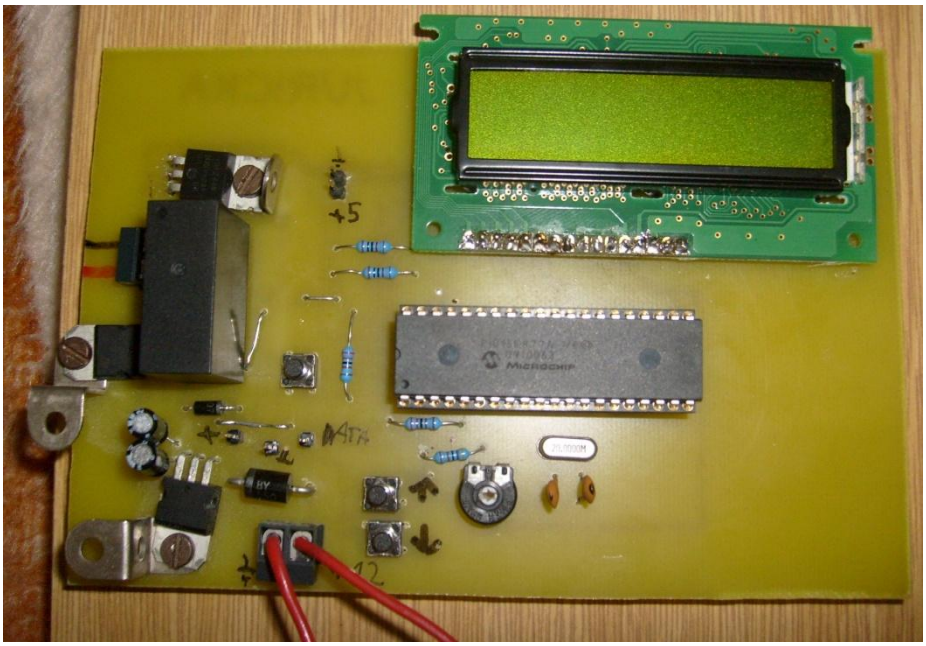
Obrázek 11: Chlazená součástka + teploměr

Z druhé strany jsem přiložil Peltierův článek a druhý chladič.

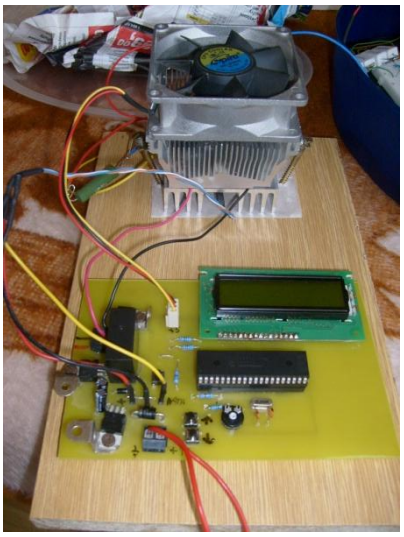


Obrázek 12: Chlazení

DPS po zapájení součástek



Obrázek 13: Deska plošných spojů + součástky



Obrázek 14: Komplet

4.2.Popis

Na prvním řádku je zobrazena změřená teplota, na druhém řádku je zobrazeno, zda se topí (H), chladí (C), nebo se nedělá nic (-). Poté je zobrazen spočítaný akční zásah a jako poslední požadovaná teplota.

Topí se na maximum.

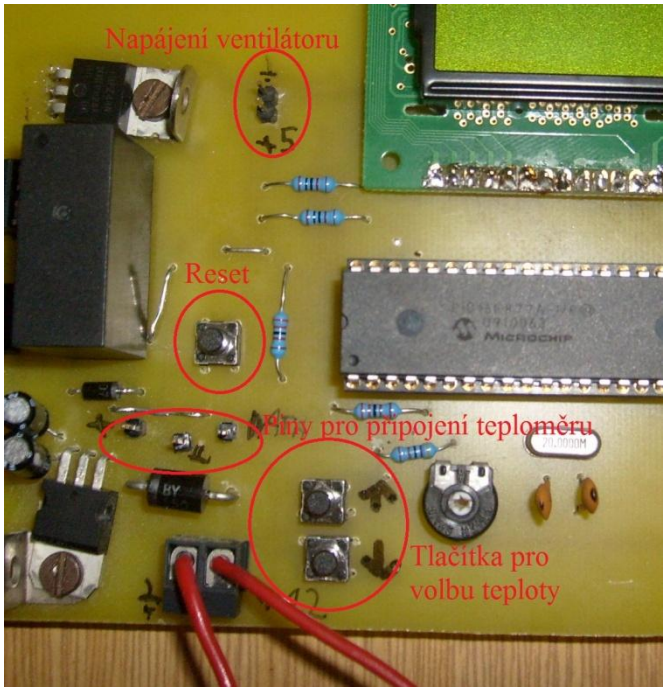
Obrázek 15: Display 1

Mělo by se chladit, ale regulátor je v režimu topení. Neprovádí se nic, čeká se na přepnutí (automaticky po cca 30 s).

Obrázek 16: Display 2

Chladí se na maximum.

Obrázek 17: Display 3



Obrázek 18: Tlačítka a rozložení pinů

5. Závěr

Podařilo se mi zkonstruovat zařízení, které je funkční a provozuschopné. Díky použití Peltierova článku je schopno chladit pod teplotu okolí, čehož by se při použití chladiče ofukovaného ventilátorem nepodařilo dosáhnout. Také je schopno vytápět, v případě, že by byla zvolena teplota vyšší, než je teplota okolí.

Při provozu bez zátěže (není přišroubovaná chlazená součástka) se dosáhne 6 °C (teplota chladiče) za 2 minuty. Poté teplota chladiče klesá velice pomalu. Dosažená teplota a rychlost by se dala zlepšit uzavřením chladiče do tepelně izolované nádoby. Zvolenou teplotu je regulátor schopen udržet v rozmezí ± 1°C. Rychlost dosažení požadované teploty lze ovlivnit správnou volbou parametrů PID regulace.

Bylo by vhodné regulátor doplnit o možnost změny parametrů bez potřeby znovu naprogramovat chip a dále naprogramovat funkci automatického nastavení parametru PID regulátoru, například některou z metod uvedenou Schlegelem [9.].

Regulátor by se také měl uzavřít do vhodné krabičky, aby byl regulátor mechanicky chráněn a obsluha chráněna před úrazem elektrickým proudem.

6. Seznam literatury:

[1.] PIC16F87XA Data Sheet [online]. Revision B. [s.l.] : Microchip Technology Inc., November 2001, October 2003 [cit. 2010-04-28]. Dostupné z WWW: <http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf>.

[2.] MC16021E8-SYL Data Sheet [online]. Wayton : Everbouquet, 15.7.2009 [cit. 2010-04-28]. Dostupné z WWW: <http://www.gme.cz/_dokumentace/dokumenty/513/513-163/dsh.513-163.2.pdf>.

[3.] MC16021E8-SERIES-3 [online]. [s.l.] : Everbouquet, [1992], 21.4.1992 [cit. 2010-04-28]. Dostupné z WWW: <http://www.gme.cz/_dokumentace/dokumenty/513/513-163/dsh.513-163.1.pdf>.

[4.] Hw.cz [online].Redakce HW serveru, 17. Únor 2000 - 1:00 [cit. 2010-04-28]. Inteligentní displeje a jejich připojení k PC | HW.cz. Dostupné z WWW: <http://hw.cz/docs/lcd_iq_displaye/lcd_iq_dip.html>.

[5.] Hw.cz [online].Redakce HW serveru, 17. Listopad 2004 - 1:00 [cit. 2010-04-28]. Sběrnice 1-Wire™ | HW.cz. Dostupné z WWW: <http://hw.cz/rozhrani/art1215-sbernice-1-wire.html>.

[6.] DS18B20 Programmable Resolution 1-Wire Digital Thermometer [online]. [s.l.] : Maxim Integrated Products, [2007], 22. 4. 2008 [cit. 2010-04-28]. Dostupné z WWW: <http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf>.

[7.] Peltierův článek In Wikipedia : the free encyclopedia [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20. 3. 2006, 19:00, 16. 1. 2010, 23:53 [cit. 2010-04-28]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Peltierův_článek>.

[8.] Hw.cz [online].Redakce HW serveru, 16. Prosinec 1999 - 1:00 [cit. 2010-04-28]. Peltierovy termobaterie | HW.cz. Dostupné z WWW: <http://hw.cz/Teorie-a-praxe/Dokumentace/ART652-Peltierovy-termobaterie.html>.

[9.] SCHLEGEL, Miloš. PRŮMYSLOVÉ PID REGULÁTORY: TUTORIAL [online]. [s.l.] : [rexcontrols], 04.03.2003 [cit. 2010-04-28]. Dostupné z WWW: <http://www.rexcontrols.cz/downloads/clanky/PIDTutor_CZ.pdf>.

[10.] Ermicroblog [online]. 30. 8. 2009 [cit. 2010-04-28]. Build Your Own Microcontroller Based PID Control Line Follower Robot. Dostupné z WWW: <http://www.ermicro.com/blog/?p=1163>.

[11.] Edaboard [online]. 2007 [cit. 2010-04-28]. Code in C for PID temperature controller. Dostupné z WWW: <http://www.edaboard.com/ftopic254719.html>.

7. Seznam tabulek

8. Seznam obrázků

9. Seznam příloh

9.1. Seznam příloh – tabulky

9.2. Seznam příloh - obrázky

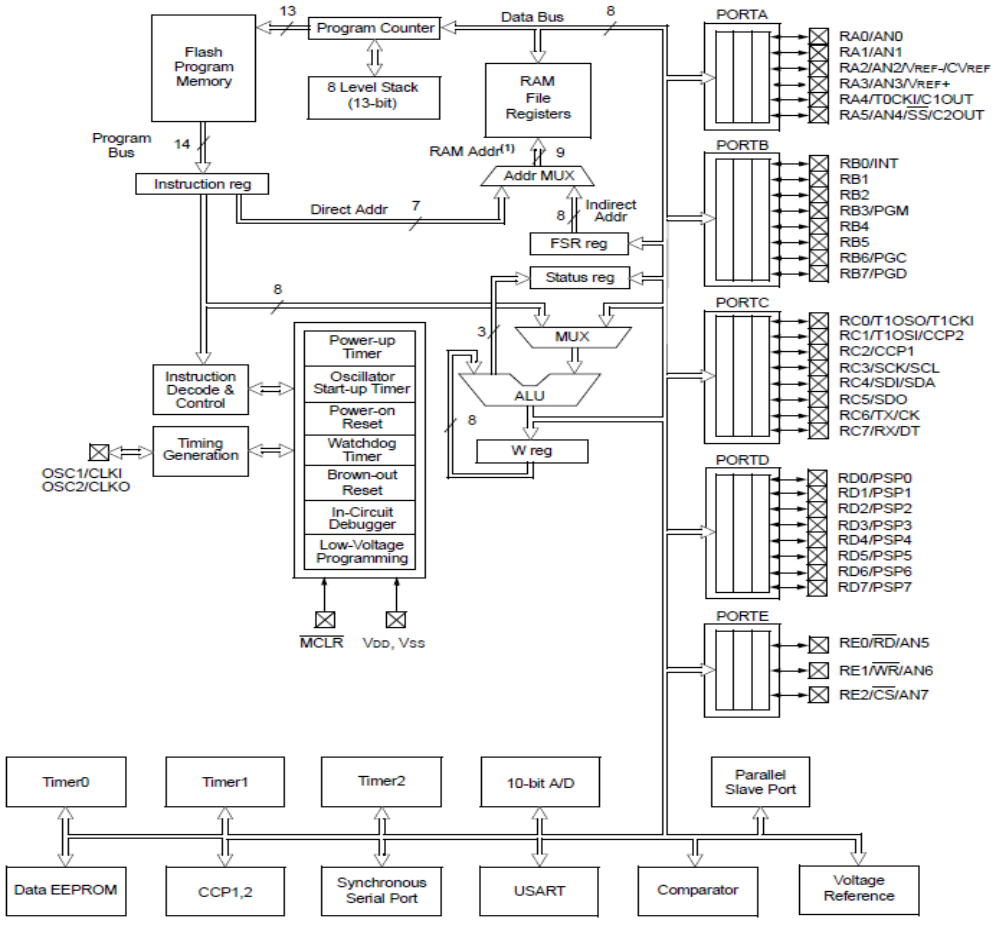
Přílohy:

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	Single Word Instructions						SPI	Master I2C			
PIC16F877A	14.3K	8192	368	256	33	8	2	Ano	Ano	Ano	2/1	2

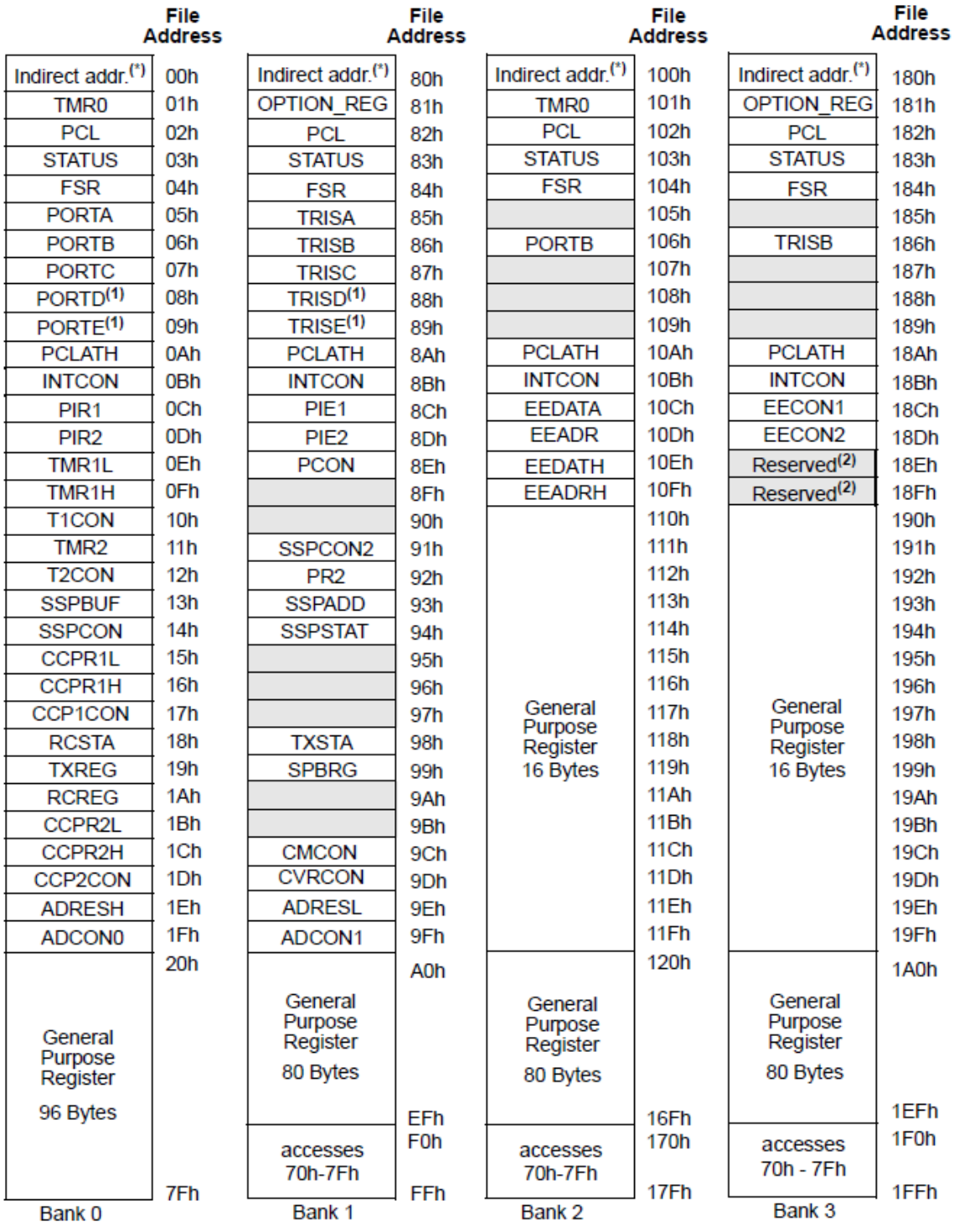
Přílohy - tabulka 1: Parametry mikroprocesoru [1.]

Key Features	PIC16F877A
Operating Frequency	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	8K
Data Memory (bytes)	368
EEPROM Data Memory (bytes)	256
Interrupts	15
I/O Ports	Ports A, B, C, D, E
Timers	3
Capture/Compare/PWM modules	2
Serial Communications	MSSP, USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
Analog Comparators	2
Instruction Set	35 Instructions
Packages	40-pin PDIP 44-pin PLCC 44-pin TQFP

Přílohy - tabulka 2: Parametry mikroprocesoru [1.]



Přílohy - obrázek 1: Blokové schéma mikroprocesoru [1.]



Přílohy - obrázek 2: Schéma paměti [1.]

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(2)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h(2)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h(2)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h(2)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah(1, 2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh(2)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	127, 150
Bank 1											
80h(2)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
82h(2)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
83h(2)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
84h(2)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150
88h	TRISD	PORTD Data Direction Register								1111 1111	48, 151
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	50, 151
8Ah(1, 2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
8Bh(2)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
8Ch	PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	29, 151
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—

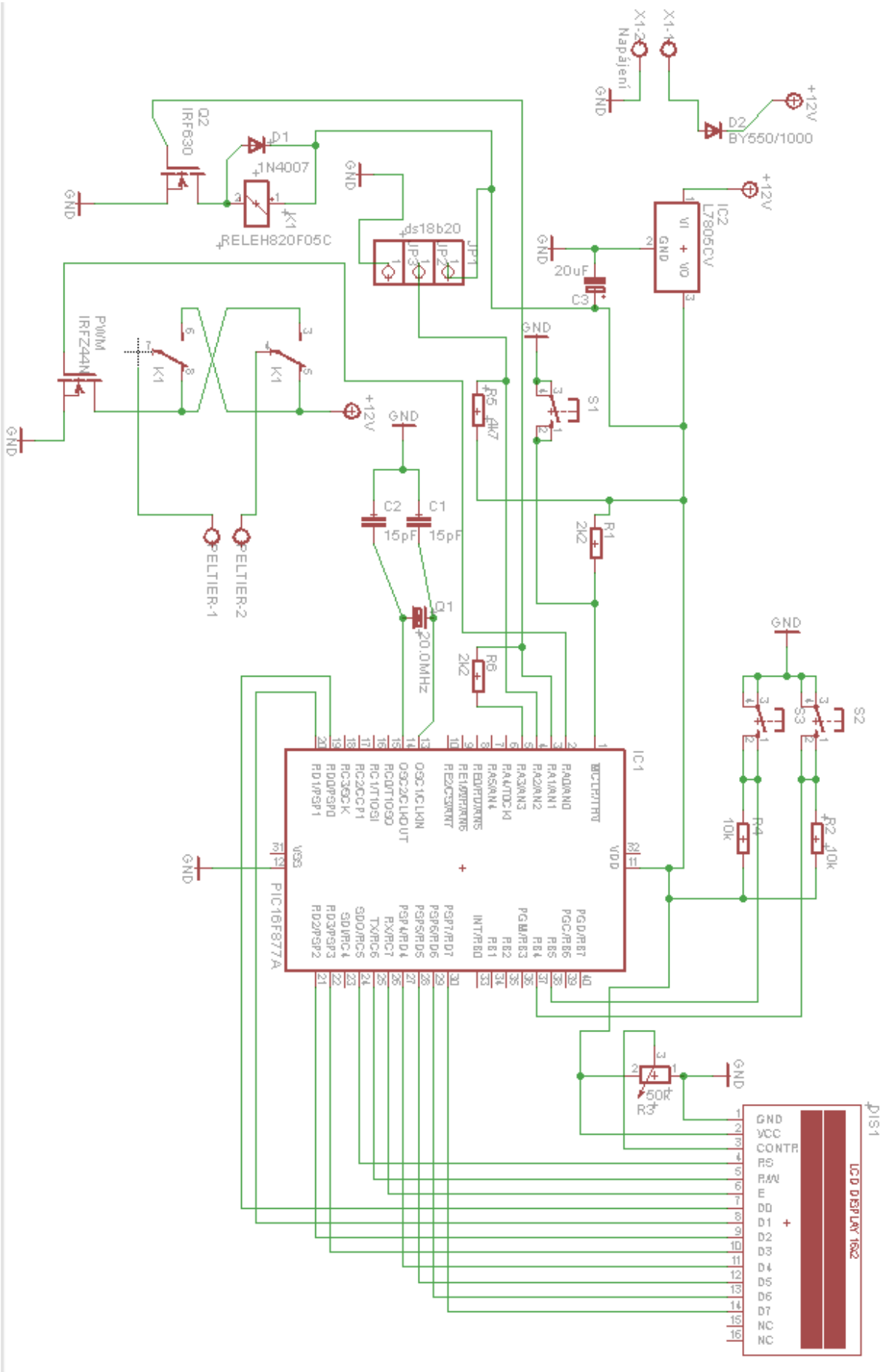
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151
92h	PR2	Timer2 Period Register								1111 1111	62, 151
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	79, 151
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	79, 151
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	133, 151
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151
Bank 2											
100h(2)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h(2)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h(2)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
104h(2)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh(2)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	39, 151
10Fh	EEADRH	—	—	—	—	EEPROM Address Register High Byte				---- xxxx	39, 151
Bank 3											
180h(2)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h(2)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h(2)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
184h(2)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,2)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh(2)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

Legenda:
x = neznámé, u = nezměněné, q = hodnota závisí na podmínkách, - = neimplementováno, čte se jako ´0´, r = reservováno
Sdílené prostory nejsou implementovány, čteny jako ´0´
Poznámka 1: Horní byte program counteru není přímo přístupný, PCLATH je udržovací registr pro PC<12:8>, jeho obsah je přenášen do horního bytu program counteru.
2: Tyto registry mohou být adresovány z kterékoli banky

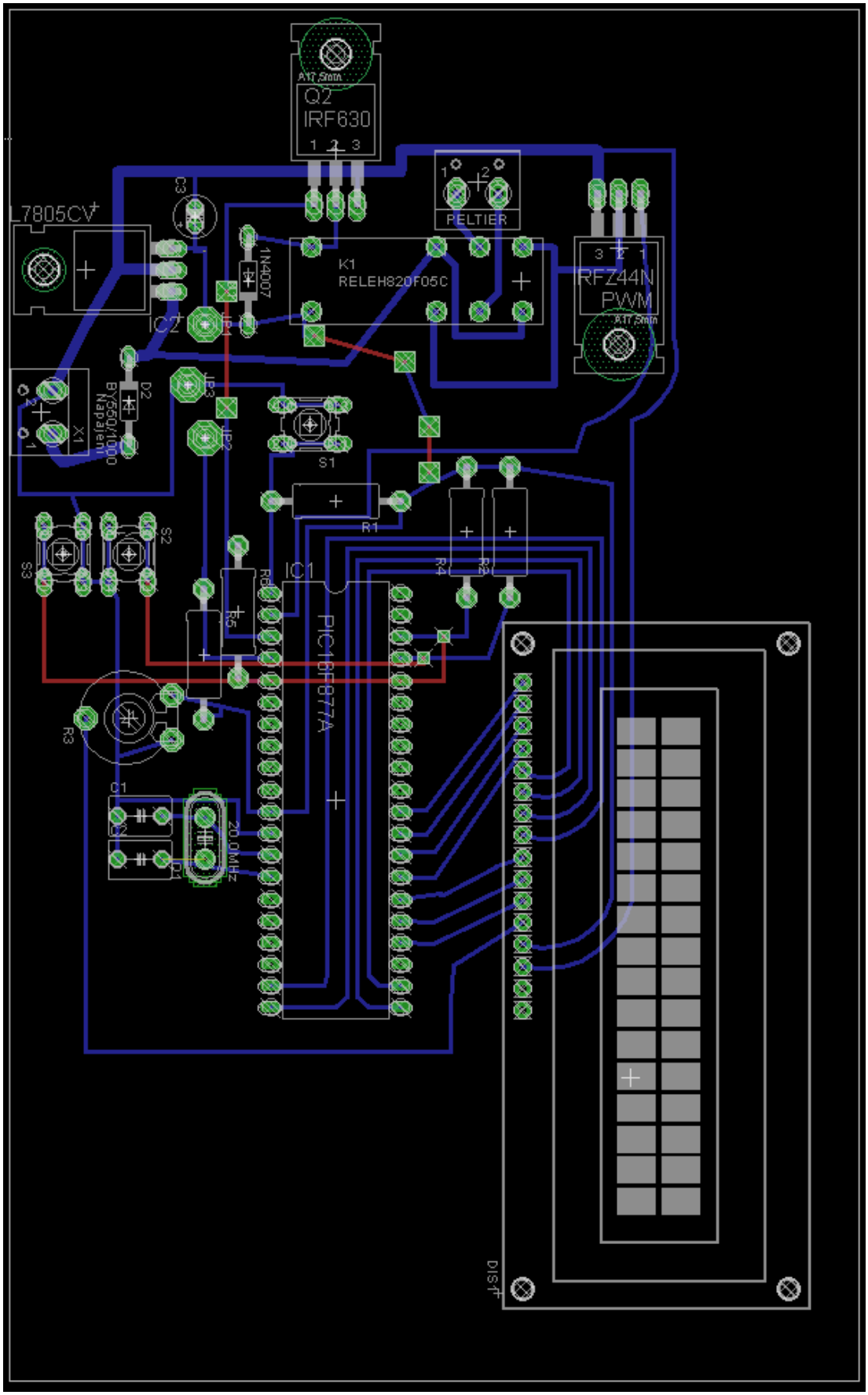
Příkaz	Kód										Popis	Čas
	RS	R / W	DB 7	DB 6	DB 5	DB 4	DB 3	DB 2	DB 1	DB 0		
Smaže displej	0	0	0	0	0	0	0	0	0	1	Smaže displej a nastaví kurzor na pozici 0.	1.64mS
Nastaví kurzor na začátek	0	0	0	0	0	0	0	0	1	*	Nastaví kurzor na pozici 0 a vynuluje posun displeje a	1.64mS
Nastaví vstupní režim	0	0	0	0	0	0	0	1	I/D	S	Určí směr pohybu kurzoru (I/D) a posun displeje (S). Tyto operace se provádějí během čtení/zápisu.	40uS
Zapne/vypne displej, kurzor a jeho blikání	0	0	0	0	0	0	1	D	C	B	Zapíná/vypíná displej (D), kurzor (C) a jeho blikání (B).	40uS
Nastaví pohyb kurzoru/displeje	0	0	0	0	0	1	S/C	R/L	*	*	Nastaví pohyb kurzoru nebo displeje (S/C) a směr pohybu (R/L). Obsah DDRAM zůstane beze změny.	40uS
Nastavení interface	0	0	0	0	1	DL	N	F	*	*	Nastaví délku interface (DL), počet řádků displeje (N) a znakový font (F).	40uS
Nastaví pozici v CGRAM	0	0	0	1	Adresa v CGRAM						Po tomto příkazu jsou data ze vstupu zaznamenávána do CGRAM namísto DDRAM.	40uS
Nastaví pozici v DDRAM	0	0	1	Adresa v DDRAM							Po tomto příkazu jsou data ze vstupu zapisována do a čtena z DDRAM.	40uS
Čte příznak BUSY a hodnotu adresového čítače	0	1	BF	DDRAM address							Čte příznak BUSY (BF) indikující, že displej ještě prování některou operaci, a pozici ukazatele adresy .	0uS
Zapíše do DDRAM nebo CGRAM.	1	0	data								Zapíše data ze vstupu DDRAM nebo do CGRAM.	40uS
Čte data z DDRAM nebo z CGRAM.	1	1	data								Čte data z aktuální adresy DDRAM nebo CGRAM.	40uS

Název bitu	Význam nastavení	
I/D	0 = Pohyb kurzoru zpět	1 = Pohyb kurzoru dopředu
S	0 = Posun displeje se neprovádí	1 = Displej se posouvá
D	0 = Vypnutí displeje	1 = Zapnutí displeje
C	0 = Vypnutí kurzoru	1 = Zapnutí kurzoru
B	0 = Vypnutí blikání kurzoru	1 = Zapnutí blikání kurzoru
S/C	0 = Pohyb kurzoru	1 = Pohyb displeje
R/L	0 = Posun doleva	1 = Posun doprava
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = střída 1/8 nebo 1/11 (jednořádkový displej)	1 = střída 1/16 (dvouřádkový displej)
F	0 = 5x7 bodů	1 = 5x10 bodů
BF	0 = Řadič přijímá instrukce	1 = Řadič je zaneprázdněn vykonáváním předchozí operace

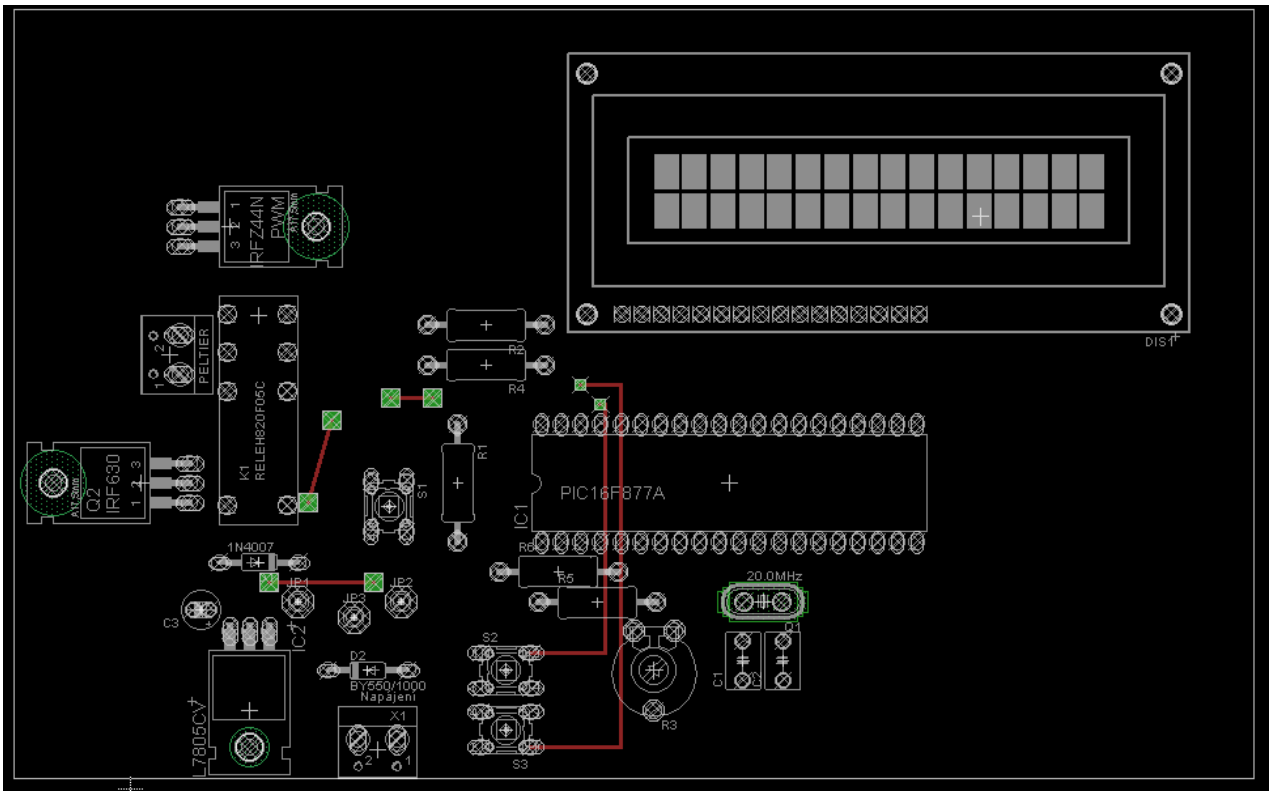
Přílohy - tabulka 4: Příkazy LCD [4.]



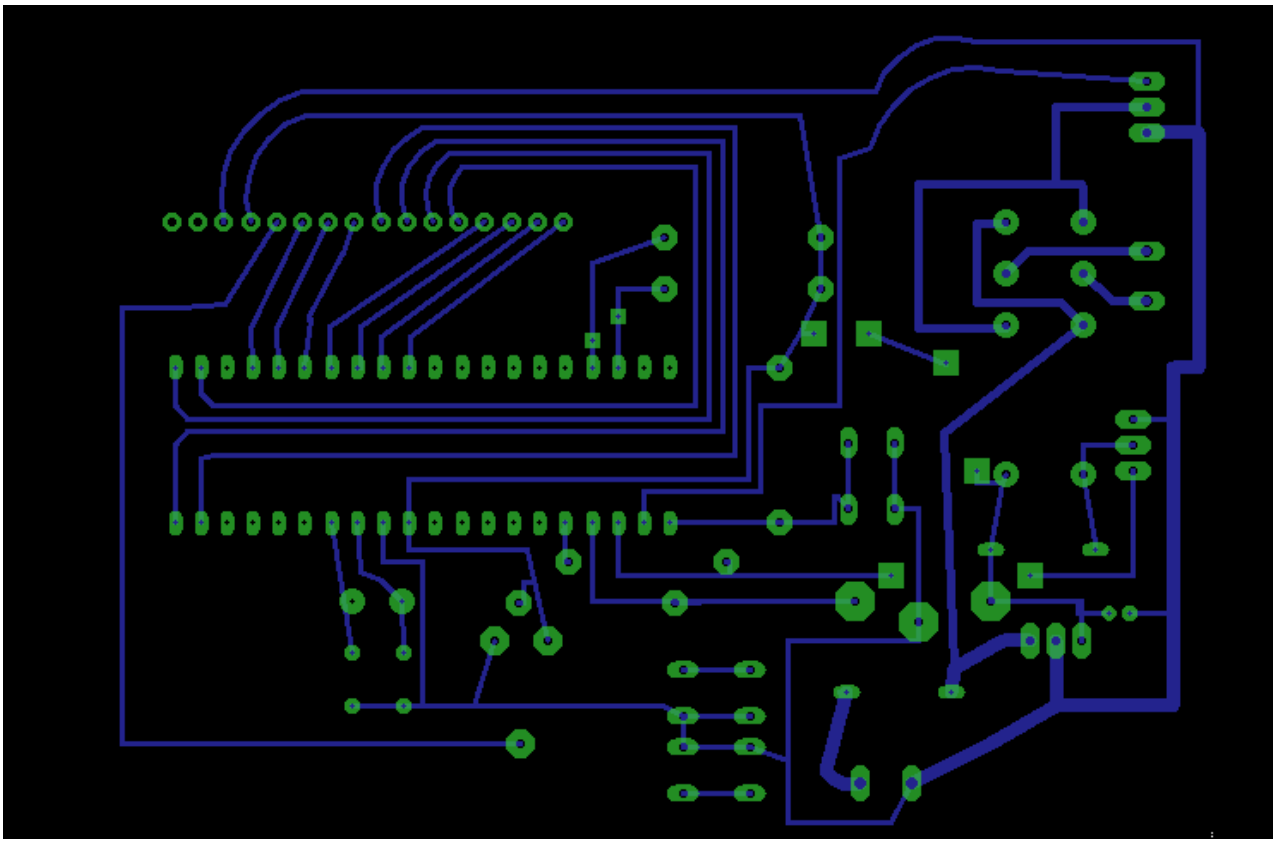
Přílohy - obrázek 3: Schéma zapojení



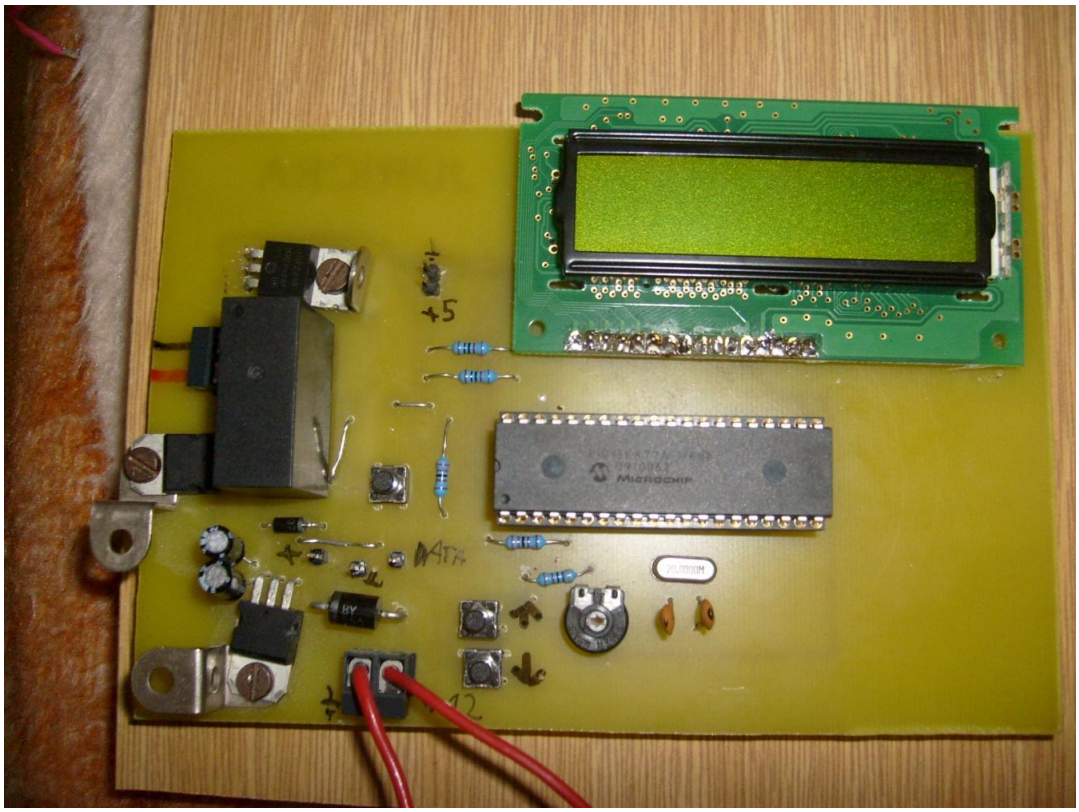
Přílohy - obrázek 4: DPS



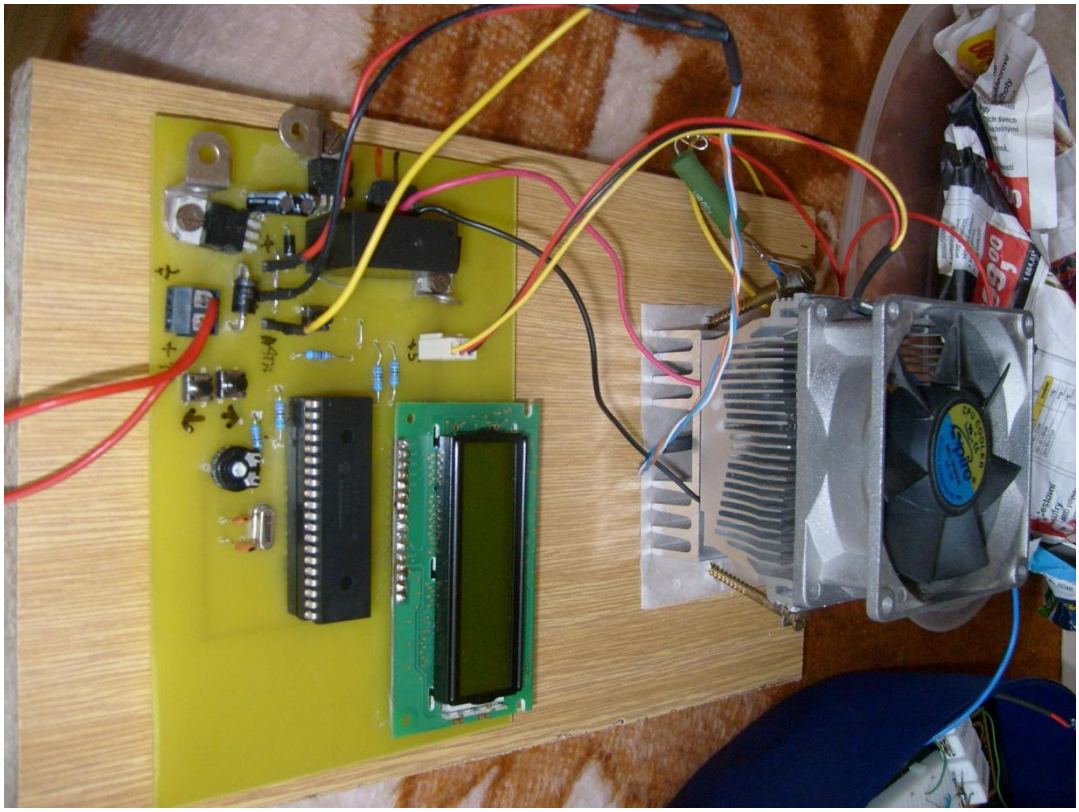
Přílohy - obrázek 5: DPS, strana součástek



Přílohy - obrázek 6: DPS, spoje



Přílohy - obrázek 7: Prototyp



Přílohy - obrázek 8: Prototyp2

Zdrojový kód:

```
/*
RA0 - výstup PWM pro spínání Peltierova článku
RA1 - výstup pro přepnutí módu ohřívání/chlazení
RA2 - vstup/výstup teploměru
RA3 - vstup pro indikaci přepnutí módu Peltierova článku
PORTD - display
RC7,6,5 - display
RB5,4 - tlačítka

*/
#include <htc.h>
__CONFIG(HS & WDTDIS & PWRTEN & BORDIS & LVPDIS & DUNPROT & WRTEN &
UNPROTECT);
#define _XTAL_FREQ 20000000
//#define bitset(var,bitno) ((var) |= 1UL < < (bitno))
//#define bitclr(var,bitno) ((var) &= ~(1UL < < (bitno)))

#include <stdio.h>
#include <stdlib.h>
#include "display.h"
#include "ds18b20.h"

signed char pozadovana=30;    //proměnná pro požadovanou teplotu
char desetiny;                //desetiny změřené teploty
unsigned int zmerena_temp=0; //změřená teplota

#define PWM_PER 255            //konstanta pro periodu PWM
unsigned char PWM_DUTY=255;   //0-255 pro střídu PWM
unsigned char PWM_PRU=0;      //počítadlo průběhu přerušením pro PWM
unsigned char PREPNI_PELTIER=0; //proměnná pro počítání k přepnutí módu
Peltierova članku
unsigned char TLACITKO_COUNT=0; //počítadlo pro ošetření zákmitu tlačítka
#define     PREPNI_CONST 30 //za jak dlouho se má přepnout mód Peltierova
článku -1 cyklus trvá cca 0,5s -> 250*0,5

void interrupt preruseni(void)
{
      if(RBIE && RBIF)//obsluha přerušení, pokud dojde k zmáčknutí
tlačítka
      {
      if(RB5==0 && pozadovana<120) pozadovana++;
      if(RB4==0 && pozadovana>-50) pozadovana--;
      TMR1ON=1;
      RBIE=0;
      RBIF=0;
      }
      if(TMR1IE && TMR1IF)//zablokování tlačítka na 210ms - ošetření
zákmitů
      {
      if(TLACITKO_COUNT<=2)
            {
            TLACITKO_COUNT++;
            }
      else
            {
            TMR1ON=0;
            TMR1H=0;
            TMR1L=0;
            RBIE=1;
```

```
            TLACITKO_COUNT=0;
            }
      TMR1IF=0;
      }

      if(TMR0IE && TMR0IF)//tmr0 používám jako časovač pro PWM
      PWM_PRU++;
      {
      if((PWM_PRU == PWM_PER))
            {
            RA0=1;
            PWM_PRU=0;
            }

      if(PWM_PRU >= PWM_DUTY)
            {
            RA0=0;
            }
      if(PWM_PRU < PWM_DUTY)
            {
            RA0=1;
            }
      TMR0IF=0;
      }
}


void BUTTONINIT()
{
ei();       // globální povolení přerušení
RBIE=1;           //povoleni přerušení od portu B
TRISB5=1;   //pin PORTB-5 jako vstup
TRISB4=1;   //pin PORTB-4 jako vstup
//*****využívám TMR1 ke generování zpoždění mezi stisky tlačítka
T1CKPS1=1;  //11 = 1:8 prescale value
T1CKPS0=1;
T1OSCEN=0;  //0= Oscillator is shut-off
TMR1CS=0;   //0= Internal clock (FOSC/4)
TMR1ON=0;   //Timer1 On bit
TMR1IE=1;   //Povolení přerušení od TMR1
PEIE=1;           //Povoleni přerušení od periferních obvodů
TMR1H=0;    //200ns*8*65536=104,85ms
TMR1L=0;
}
//********** interní PWM modul, nastaveni PWM, nepoužívá se, problémy při
vysoké frekvenci spínání velké zátěže - peltier
void PWMINIT()
{
      //PWM peltier-chlazeni, topení
      TMR2ON=0; //Timer2 On bit -> 0ff
      TRISC2=0;   //PortC2    jako vystup(vystup PWM)
      TRISC1=0;
      PR2=0xff;   //Timer2 Period Register - f=1.22 kHz
      TMR2IE=0;   //zakázání přerušení od TMR2
      CCP1IE=0;
      CCPR1L=0;


      TOUTPS3=0;  //Timer2 Output Postscale Select bits -> 1:1
      TOUTPS2=0;
      TOUTPS1=0;
```

```
      TOUTPS0=0;
      T2CKPS1=1;  //Timer2 Clock Prescale Select bits -> 1
      T2CKPS0=0;

      CCP1X=0;    //WM duty cycle -> 2LSBPWM Least Significant bits
      CCP1Y=0;
      CCP1M3=1;   //mode select -> PWM mode
      CCP1M2=1;
      CCP1M1=0;
      CCP1M0=0;
      TMR2ON=1;   //    Timer2 On bit -> 0n
//CCPR1L:CCP1CON<5:4>) 0-800 == 0-100% PWM
}

#define MAX_respond 255       //omezení maximální odezvy regulátoru
#define MIN_respond -255      //omezení minimální odezvy regulátoru
float       error;//e0
float predchozi_error;  //e1
float I;                            //integrační složka
float D;                            //derivační složka
#define Kp 400.0              //konstanta proporcialni složky t=1 kp=200
#define Ti 20.0               //konstanta integralni složky
//chlazeni
//      rozptyl[°C]
//ti  dolu      nahoru
//40  0.3         0.4   t1    kp200
//35  0.3         0.2   t1    kp200
//30  0.3         0.3   t0.5  kp400
//25  0.2         0.4   t0.5  kp400
//20  0.2         0.5
//20  0.2         0.2   t0.5  kp400 td1
//30  0.1         0.3   t0.5  kp400 td5
//30  0.2         0.3   t0.5  kp400 td1
//30  0.1         0.2   t0.5  kp400 td10 déle mu trvá ustálení
//30  0.2         0.3   t0.5  kp400 td3
//20  0.1         0.2   t0.5  kp400 td3 //asi nejlepší nastaveni



#define Td 3.0                      //konstanta derivační složky
float vysledek=0.0;
#define T 0.5                       //vzorkovací čas time
const float qd=Kp*Td/T;
const float qi=Kp*T/Ti;
int vysledek_int;             //výsledek převedený na int, aby nemuselo
docházet k přetypování proměnných


void pid( )
{

      error=pozadovana-(zmerena_temp+(desetiny*0.0625));
      if
(!(((vysledek<=MIN_respond)&&(error<0))||((vysledek>=MAX_respond)&&(error
>0)))) //antiwindup integral - ochrana proti růstu integrační složky,
pokud je výsledek v saturaci
      {
            I=I+qi*error;
      }
      D=qd*(error-predchozi_error);
      vysledek=Kp*error+I+D;        //výsledný akční zásah
```

```
      if (vysledek>MAX_respond) vysledek=MAX_respond; //omezení na
maximální odezvu které jsme schopni dosáhnout
      if (vysledek<MIN_respond) vysledek=MIN_respond; //omezení na
minimální odezvu které jsme schopni dosáhnout

      predchozi_error=error;  //uložení posledního eroru do dalšího cyklu
      vysledek=round(vysledek);     //zaokrouhlení výsledku
      vysledek_int=(int)vysledek;

      LCDSETPOZITION(0x40);   //přesun na vhodnou pozici na displeji

      if(vysledek_int==0)
      {
      //netopit, nechladit;
            PWM_DUTY=40;       //na Peltierovém článku by se mělo nechat
něco málo, protože když je úplně vypnut, tak je ideální vodič
tepla(teplota na obou stranách Peltierova článku se vyrovná)
            PREPNI_PELTIER=0; //netřeba přepínat chlazeni/topeni
      }
      else
      {
            if(vysledek_int<0) //chladit
            {
                  if(RA3==0)//RA1==0->chladící mod
                  {
                        PWM_DUTY=-vysledek_int;
                        LCDPUTSTRING("C ");     //signalizace chlazení
                        PREPNI_PELTIER=0; //vynulování konstanty pro
přepnuti-je v chladícím modu a má chladit
                  }
                  else
                  {
                        PWM_DUTY=0;
                        LCDPUTSTRING("- "); //signalizace nechladí/netopí
                        PREPNI_PELTIER++; //je v ohřívacím módu, ale má
chladit
                  }
            }
            else //vysledek_int>0
            {
            //topit
                  if(RA3==0)
                  {
                        PWM_DUTY=0;
                        LCDPUTSTRING("- "); //signalizace nechladí/netopí
                        PREPNI_PELTIER++; //je v chladícím modu, ale má
topit
                  }
                  else
                  {
                        PWM_DUTY=vysledek_int;
                        LCDPUTSTRING("H ");
                        PREPNI_PELTIER=0; //vynulováni konstanty pro
přepnuti-je v ohřívacím modu a má ohřívat
                  }
            }
      }
      if(PREPNI_PELTIER>=PREPNI_CONST)
      {
            if(RA1==0)
                  RA1=1;
```

```
            else
                  RA1=0;
      }
}

//*********************************************************************
***************************
void PWMINIT2()
{
//použití s tmr1
/*T1CKPS1=1; //11 = 1:8 prescale value
T1CKPS1=0;
T1OSCEN=0; //0= Oscillator is shut-off
TMR1CS=0; //0= Internal clock (FOSC/4)
TMR1ON=0; //Timer1 On bit
TMR1IE=1;
PEIE=1;
TMR1H=0;//přerušení vyvolané každých 1,6384ms
TMR1L=0;
TRISA0=0;//vystup PWM
RA0=1;
TMR1ON=1; //Timer1 On bit*/

// pwm s tmr0
T0CS=0;     //0 = Internal instruction cycle clock (CLKO)
PSA=1;            //0= Prescaler is assigned to the Timer0 module
PS2=0;            //Prescaler Rate Select bits - TMR0 Rate 1 : 256
PS1=0;
PS0=0;
TMR0=0;           //vynulování timeru
TMR0IE=1;   //povolení přerušení od TMR0
GIE=1;            //globální povolení přerušení
TRISA0=0;   //PORT A0 jako vystup PWM
RA0=1;
//perioda PWM_PER(255)*4*Tosc*256=13ms=76.6Hz
}
/*********************************************
 *FUNKCE MAIN
*********************************************/
void main(void)
{
      ADCON1=0x06;      //porta jako TTL i/o
      TRISA1=0;         //vystup pro přepínání modu chlazení/ohřívání
RA1==0-chlazení
      TRISA3=1;         //vstup pro signalizaci přepnutí módu
      RA1=0;            //chladící mód jako výchozí
      LCDINIT();        //podprogram pro inicializaci LCD
      BUTTONINIT();     //podprogram pro inicializaci tlačítek
      PWMINIT2();       //podprogram pro inicializaci PWM


      SENDPRIKAZ(0x01); //vynulování displeje a přesun na první pozici


      do{
            LCDSETPOZITION(0);            //nastav pozici
            LCDPUTSTRING("Temp:");        //vypiš na display
            zmerena_temp=GETTEMP();       //změř teplotu
//***********převod a zobrazeni na display
            desetiny=(char)(zmerena_temp&0x000f);
            zmerena_temp>>=4;
```

```
            (signed char)zmerena_temp;
            if(zmerena_temp<0.0) desetiny=(~desetiny&0x0f)+1;

            WRITENUMBER(zmerena_temp);          //zapiš změřenou teplotu
            WRITEDESETINY(desetiny*625);        //desetiny teploty
            LCDPUTCHAR(0xdf);                   //°
            LCDPUTCHAR('C');                    //C
            pid();                  // vypočítej a proveď akční zásah
//*************************************
            LCDSETPOZITION(0x42);               // nastav pozici
            WRITENUMBER(vysledek_int);          //vypiš akční zásah
            LCDPUTSTRING("Req:");               //vypiš na display
            WRITENUMBER(pozadovana);            //požadována teplota
            LCDPUTCHAR(0xdf);                   //°
            LCDPUTCHAR('C');                    //C
      }
      while(1);   //měření a regulace stále do kola
}
/***********************************
PROCEDURY PRO DISPLAY***************
************************************
!!!!!display připojen na port D(8/bit) komunikace, portC5-7!!!!!
*/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define     display     PORTD
#define rs  RC5
#define rw  RC6
#define e   RC7
#define     output      0
#define input     1


void LCDBUSY()    //zjistí, jestli je display zaneprázdněn
{
      TRISD=0xff;
      do{
            e=0;
            rs=0;
            rw=1;
            e=1;
      }while((display&0b10000000)==128);
      e=0;
      rw=0;
      TRISD=0x00;
}

void SENDPRIKAZ_NO_WAIT(unsigned char data)//pošle příkaz na display bez
zjišťování, zda zrovna něco neprovádí
{
      rs=0;
      rw=0;
      e=1;
      display=data;
      __delay_us(1);
      e=0;
}
```

```
//jakmile bude display připraven, pošle příkaz
void SENDPRIKAZ(unsigned char data)
{
   LCDBUSY(); //Čekání na display až bude připraven
   SENDPRIKAZ_NO_WAIT(data);
}

//pošle jeden znak na aktuální pozici
void LCDPUTCHAR(unsigned char znak)
{
      LCDBUSY(); // Čekání na display až bude připraven
      rs=1;
      rw=0;
      e=1;
      display=znak;
      e=0;

}

//nastaví pozici na displeji
void LCDSETPOZITION(unsigned char pozice)
{
      LCDBUSY();
      rs=0;
      rw=0;
      e=1;
      display=(pozice|0b10000000);
      e=0;
}

//zjistí aktuální pozici na displeji
unsigned char LCDGETPOZITION()
{
      unsigned char pozice;
      TRISD=0xff;
      e=0;
      rs=0;
      rw=1;
      e=1;
      pozice=display&0b01111111;
      e=0;
      rw=0;
      TRISD=0x00;
      return pozice;
}

//napíše text od aktuální pozice
void LCDPUTSTRING(const char * s)
{
 while(*s)
 LCDPUTCHAR(*s++);
}

//prevede cislo na BCD format a posle na display na aktualni pozici
void WRITENUMBER(signed int cislo)
{
      if(cislo < 0) {
            LCDPUTCHAR('-');
            cislo=0-cislo;
      }
      else
```

```
	LCDPUTCHAR(' ');

	LCDPUTCHAR((cislo/100)+0x30);
	cislo=fmod(cislo,100);
	LCDPUTCHAR((cislo/10)+0x30);
	cislo=fmod(cislo,10);
	LCDPUTCHAR(cislo+0x30);
}

//převede číslo na BCD formát a zapíše na display
void WRITEDESETINY(int cislo)
{
	LCDPUTCHAR(',');
	LCDPUTCHAR((cislo/1000)+0x30);
	cislo=fmod(cislo,1000);
	if(cislo==0)
		{
		LCDPUTSTRING("000");
		return;
		}
	LCDPUTCHAR((cislo/100)+0x30);
	cislo=fmod(cislo,100);
	if(cislo==0)
		{
		LCDPUTSTRING("00");
		return;
		}
	LCDPUTCHAR((cislo/10)+0x30);
	cislo=fmod(cislo,10);
	if(cislo==0)
		{
		LCDPUTSTRING("0");
		return;
		}
	LCDPUTCHAR(cislo+0x30);
}

//inicializace komunikace s displejem
void LCDINIT()
{
	TRISD=output;
	TRISC7=output;
	TRISC6=output;
	TRISC5=output;
	__delay_ms(15);
	SENDPRIKAZ_NO_WAIT(0b00111111);
	__delay_ms(4);
	__delay_us(100);
	SENDPRIKAZ_NO_WAIT(0b00111111);
	__delay_us(100);
	SENDPRIKAZ_NO_WAIT(0b00111111);
	__delay_us(40);
	SENDPRIKAZ(0b00111000); //Nastaví délku interface (8bit), počet
řádků displeje (2) a znakový font ( 5x7 bodů).
	SENDPRIKAZ(0b00001000); //Zapíná/vypíná displej (vypnut), kurzor
(vypnut) a jeho blikání (vypnuto).
	SENDPRIKAZ(0b00000001); //Smaže displej a nastaví kurzor na pozici
0.
	SENDPRIKAZ(0b00000110); //Určí směr pohybu kurzoru (Zvýšení pozice
kurzoru) a posun displeje (displ. nepohybovat). Tyto operace se provádějí
během čtení/zápisu.
```

```
      SENDPRIKAZ(0b00001100); //Zapíná/vypíná displej (zapnut), kurzor
(vypnut) a jeho blikání (vypnuto).
}
/*********************************
Procedury pro teploměr DS18B20
*********************************/
#define read RA2
#define RX TRISA2=1

void TX()
{
TRISA2=0;   //trisc4 výstup
RA2=0;
}

// detekce přítomnosti teploměru
unsigned char PRESENCEPULSE()
{
      unsigned char navratovaa=1;     // výchozí návratová hodnota

      GIE=0;                           //zakázání přerušení
      TX();                           // bus low
      __delay_us(490);                // čas pro příkaz reset
      RX;
      __delay_us(70);                 // čekání na potvrzeni teploměrem
      if(read) navratovaa=0;          // pokud detekována log.1, tak
teploměr na sbernici neni
      GIE=1;                                          //povolení přerušení
      __delay_us(420);                // pauza před další komunikací
      return navratovaa;              // vrátí stav 1=teploměr nalezen,
0=teploměr nenalezen
}

void WRITEZERO()
{
      GIE=0;               //zakázání přerušení
      TX();               // bus low
      __delay_us(70);     // pauza definující log.0
      RX;                 // uvolnění sběrnice
    GIE=1;                //povolení přerušení
      __delay_us(10);     // pauza před další komunikací
}

void WRITEONE()
{
      GIE=0;              //zakázání přerušení
      TX();               // bus low
      __delay_us(4);      // pauza definující log.1
      RX;                 // uvolnění sběrnice
      GIE=1;              //povolení přerušení
      __delay_us(76);     // pauza před další komunikací
}

unsigned char READ()
{
      unsigned char navratova=0;    // výchozí návratová hodnota bitu
      GIE=0;                         //zakázání přerušení
      TX();                         // bus low
      __delay_us(2);                // pauza pro stav čtení
      RX;                           // uvolnění sběrnice
      __delay_us(3);                // pauza pro reakci teploměru
```

```
      if(read) navratova=1;       // test stavu sběrnice - vlastní čtení
      GIE=1;                                    //povolení přerušení
      __delay_us(65);             // pauza před další komunikací
      return navratova;           // přečtená hodnota, 1 nebo 0
}

//funkce pro zápis celého bytu
void WRITEBYTE(unsigned char zapis)
{
      unsigned char i=0;
      unsigned char tmp_zapis=zapis;
      for(i;i<8;i++)
      {
            if(tmp_zapis&1)
                  WRITEONE();
            else
                  WRITEZERO();
            tmp_zapis>>=1;
      }
}

//funkce pro přečtení bytu
unsigned char READBYTE()
{
      volatile unsigned char navrat=0;
      volatile unsigned char i=0;
      for(i;i<8;i++)
      {
            navrat >>=1;
            if(READ()) navrat|=0x80;
      }
      return navrat;
}

//funkce co vyšle příkaz pro změření teploty, přečte ji a vrátí
unsigned int GETTEMP()
{
      unsigned int teplota=0;
      unsigned char teplota_lo=0, teplota_hi=0;
      while(PRESENCEPULSE()==0)continue;
      WRITEBYTE(0x0CC);             //přeskoč identifikaci
      WRITEBYTE(0x044);             //změř teplotu
      while(READ()==0)continue; //čekání na převod teploty skončeno
read==1
      while(PRESENCEPULSE()==0)continue;
      WRITEBYTE(0x0CC);             //přeskoč identifikaci
      WRITEBYTE(0x0BE);             //pošli naměřené hodnoty
      teplota_lo=READBYTE();
      teplota_hi=READBYTE();
      teplota=teplota_hi;
      teplota <<= 8;
      teplota|=teplota_lo;
      PRESENCEPULSE();
      return teplota;
}
```